

AN EFFICIENT & ROBUST RIDESHARING FARE PREDICTION SYSTEM USING NEURAL NETWORK

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering Degree in Computer Science and
Engineering by

TATICHARLA DURGA PRASAD (Reg. 37110203)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI

SALAI, CHENNAI – 600119.

APRIL 2021



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade “A” by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
Chennai - 600119
www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that TATICCHARLA DURGA PRASAD (Reg. 37110203) has done this Project Report is the bonafide work of who carried out the project entitled “**AN EFFICIENT & ROBUST RIDESHARING FARE PREDICTION SYSTEM USING NEURAL NETWORK**” under our supervision from November 2020 to April 2021.

Internal Guide

Dr.A.Viji Amutha Mary,M.Tech.,Phd.,

Head of the Department

Dr. S. VIGNESHWARI M.E., Ph.D.

Dr. L. LAKSHMANAN M.E.,Ph.D.,

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I, **TATICHARLA DURGA PRASAD (37110203)** hereby declare that the Project Report entitled “ **AN EFFICIENT & ROBUST RIDESHARING FARE PREDICTION SYSTEM USING NEURAL NETWORK**” done by me under the guidance of **Dr.A.Viji Amutha Mary**, at Sathyabama Institute of Science and technology is submitted in partial fulfillment of the requirements of the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to Board of management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

We convey our thanks to **Dr. T. SASIKALA M.E., Ph.D., Dean, School of Computing, Dr. S. VIGNESHWARI M.E., Ph.D. and Dr. L. LAKSHMANAN M.E., Ph.D., Heads of the Department,** Department of Computer Science and Engineering for providing us the necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Dr.A.Viji Amutha Mary,M.Tech.,Phd.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the Department of **COMPUTER SCIENCE AND ENGINEERING** who were helpful in many ways for the completion of the project.

ABSTRACT

Ridesharing, a shared service that uses the information and knowledge matching, can efficiently utilize scattered social resources to reduce the demand for vehicles in urban road networks. However, car ridesharing has the problems of low capacity and high cost, and it cannot satisfy demands for recurring, long-distance, and low-cost trips. In this paper, we formally define the bus ridesharing problem and propose a large-scale bus ridesharing service to resolve this problem. The proposed system exploits the mobility information of ride requests and taxis to achieve efficient indexing of taxis/requests and better passenger-taxi matching, while still satisfying the constraints on passengers' deadlines and taxis' capacities. Specifically, the proposed system indexes taxis and ride requests with both geographical information and travel directions, and supports the shortest path based routing and probabilistic routing to serve both online and offline ride requests. Extensive experiments with a large real-world taxi dataset demonstrate the efficiency and effectiveness of the proposed system, which can response each ride request in milliseconds and with a moderate detour cost.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NUMBER
	ABSTRACT	vi
	LIST OF FIGURES	viii
1	INTRODUCTION	1
	1.1 OVERVIEW	1
	1.2 SCOPE OF THE PROJECT	1
	1.3 DOMAIN OVERVIEW	1
	1.3.1 MACHINE LEARNING	2
2	LITERATURE SURVEY	7
3	SYSTEM ANALYSIS	10
	3.1 EXISTING SYSTEM	15
	3.2 DRAWBACKS OF EXISTING SYSTEM	16
	3.3 PROPOSED SYSTEM	16
	3.4 ADVANTAGES OF PROPOSED SYSTEM	17
4	REQUIREMENT SPECIFICATION	18
	4.1 INTRODUCTION	18
	4.2 HARDWARE & SOFTWARE SPECIFICATION	18
	4.2.1 HARDWARE SPECIFICATION	18
	4.2.2 SOFTWARE SPECIFICATION	19
	4.3 SOFTWARE DESCRIPTION	19
	4.3.1 ABOUT PYTHON	19
	4.3.2 ABOUT IDE	30
5	SYSTEM DESIGN	35
	5.1 INTRODUCTION	35

	5.2 ARCHITECTURE DIAGRAM	35
6	SYSTEM IMPLEMENTATION	37
	6.1 LIST OF MODULES	37
	6.1.1 EXPLORATORY OF DATA EVALUATION	37
	6.1.2 PRE-PROCESSING	37
	6.1.3 FEATURE ENGINEERING	38
	6.1.4 PREDUCTION	39
7	SOFTWARE TESTING	41
	7.1 GENERAL	46
	7.2 TEST DRIVEN DEVELOPMENT	47
	7.3 UNIT TESTING	48
	7.4 BLACKBOX TESTING	49
	7.5 INTEGRATION TESTING	50
	7.6 SYSTEM TESTING	51
	7.7 SANITY TESTING	53
	7.8 REGRESSION TESTING	54
8	CONCLUSION & FUTURE ENHACEMENT	58
	8.1 CONCLUSION	58
	8.2 FUTURE ENHANCEMENT	58
	APPENDIX-I SCREEN CODING	59
	APPENDIX-II SAMPLE SHOTS	72
	REFERENCE	80

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
1.1	MACHINE LEARNING	3
3.1	WATERFALL MODEL	19
3.2	RAPID APPLICATION DEVELOPMENT	20
4.1	PYTHON OPERATING SYSTEM	28
4.2	PYTHON VISUALIZATION TOOL	28
5.1	ARCHIETURE DIAGRAM	44
7.1	SOFTWARE DEVELOPMENT CYCLE	53
7.2	TESTING FRAMEWORKS	55
7.3	INTEGRATION TESTING	59
7.4	SYSTEM TESTING	59
7.5	REGRESSION TESTING	64

CHAPTER 1

INTRODUCTION

A novel peer-to-peer system for matching car rides based on their routes and travel times. Unlike other ridesharing systems, Ride Matcher is completely decentralized, which makes it possible to deploy it on distributed infrastructures

1.1 OVERVIEW

Some orders might be failed to match a suitable vehicle due to the constraint of maximum delay. In order to improve the success rate, we introduce a waiting module with a first-in first out scheme to prioritize those passengers who have waited for a long time. The system will serve the order with earlier birth time first in each processing period. To save the calculation resource, the orders in the waiting list will be retrieved only once.

1.2 SCOPE OF THE PROJECT

The main contributions of this project therefore are:

- Data Analysis
- Dataset preprocessing
- Training the Model
- Testing of Dataset

1.3 DOMAIN OVERVIEW

Machine Learning is the most popular technique of predicting the future or classifying information to help people in making necessary decisions. Machine Learning algorithms are trained over instances or examples through which they learn from past experiences and also analyze the historical data. Therefore, as it trains over the examples, again and again, it is able to identify patterns in order to make predictions about the future.

Data is the core backbone of machine learning algorithms. With the help of the historical data, we are able to create more data by training these machine

learning algorithms. For example, Generative Adversarial Networks are an advanced concept of Machine Learning that learns from the historical images through which they are capable of generating more images. This is also applied towards speech and text synthesis. Therefore, Machine Learning has opened up a vast potential for data science applications.

1.3.1 MACHINE LEARNING

Machine Learning combines computer science, mathematics, and statistics. Statistics is essential for drawing inferences from the data. Mathematics is useful for developing machine learning models and finally, computer science is used for implementing algorithms.

However, simply building models is not enough. You must also optimize and tune the model appropriately so that it provides you with accurate results. Optimization techniques involve tuning the hyperparameters to reach an optimum result.

The world today is evolving and so are the needs and requirements of people. Furthermore, we are witnessing a fourth industrial revolution of data. In order to derive meaningful insights from this data and learn from the way in which people and the system interface with the data, we need computational algorithms that can churn the data and provide us with results that would benefit us in various ways. Machine Learning has revolutionized industries like medicine, healthcare, manufacturing, banking, and several other industries. Therefore, Machine Learning has become an essential part of modern industry.

Data is expanding exponentially and in order to harness the power of this data, added by the massive increase in computation power, Machine Learning has added another dimension to the way we perceive information. Machine Learning is being utilized everywhere. The electronic devices you use, the applications that are part of your everyday life are powered by powerful machine learning algorithms.

With an exponential increase in data, there is a need for having a system that can handle this massive load of data. Machine Learning models like Deep Learning allow the vast majority of data to be handled with an accurate generation of

predictions. Machine Learning has revolutionized the way we perceive information and the various insights we can gain out of it.

These machine learning algorithms use the patterns contained in the training data to perform classification and future predictions. Whenever any new input is introduced to the ML model, it applies its learned patterns over the new data to make future predictions. Based on the final accuracy, one can optimize their models using various standardized approaches. In this way, Machine Learning model learns to adapt to new examples and produce better results.

Types of Machine Learning

Machine Learning Algorithms can be classified into 3 types as follows

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

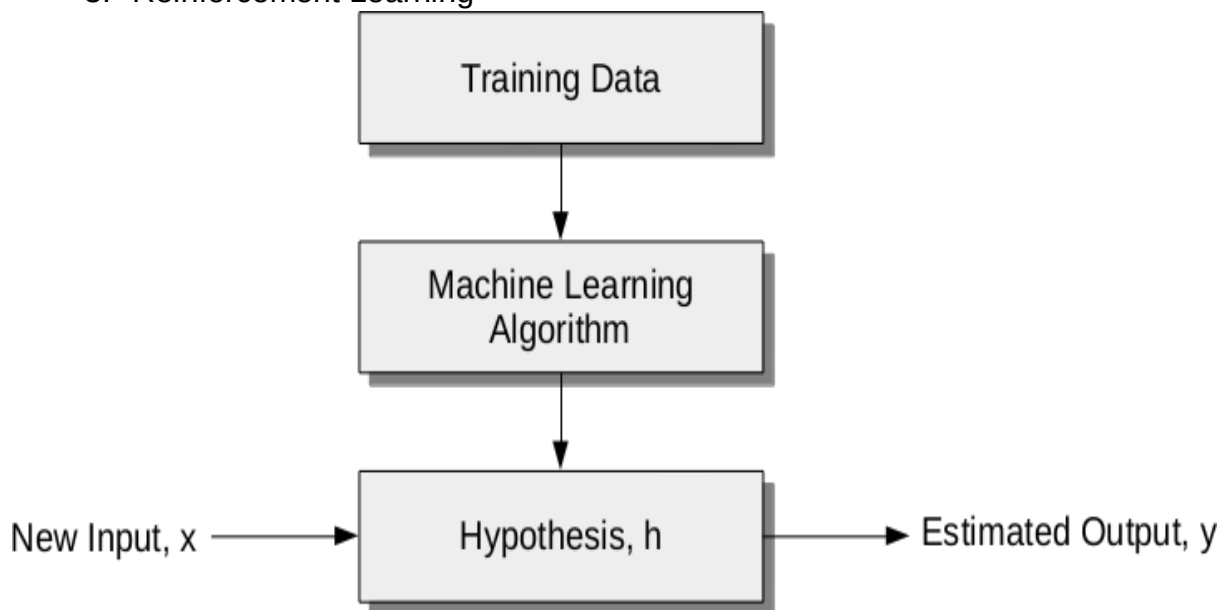


Fig 1.1

1.41.4

Supervised Learning

In the majority of supervised learning applications, the ultimate goal is to develop a finely tuned predictor function $h(x)$ (sometimes called the “hypothesis”). “Learning” consists of using sophisticated mathematical algorithms to optimize this function so that, given input data x about a certain domain (say, square footage of a house), it will accurately predict some interesting value $h(x)$ (say, market price for said house).

$$h(x_1, x_2, x_3, x_4) = \theta_0 + \theta_1 x_1 + \theta_2 x_3^2 + \theta_3 x_3 x_4 + \theta_4 x_1^3 x_2^2 + \theta_5 x_2 x_3^4 x_4^2$$

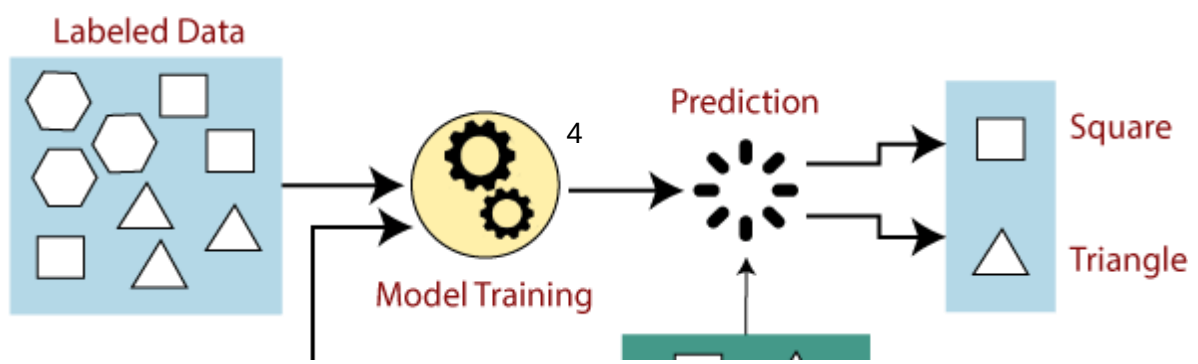
This function takes input in four dimensions and has a variety of polynomial terms.

Deriving a normal equation for this function is a significant challenge. Many modern machine learning problems take thousands or even millions of dimensions of data to build predictions using hundreds of coefficients. Predicting how an organism’s genome will be expressed, or what the climate will be like in fifty years, are examples of such complex problems.

Under supervised ML, two major subcategories are:

- Regression machine learning systems: Systems where the value being predicted falls somewhere on a continuous spectrum. These systems help us with questions of “How much?” or “How many?”.
- Classification machine learning systems: Systems where we seek a yes-or-no prediction, such as “Is this tumor cancerous?”, “Does this cookie meet our quality standards?”, and so on.

In practice, x almost always represents multiple data points. So, for example, a housing price predictor might take not only square-footage (x_1) but also number of bedrooms (x_2), number of bathrooms (x_3), number of floors (x_4), year built (x_5), zip code (x_6), and so forth. Determining which inputs to use is an important part of ML design. However, for the sake of explanation, it is easiest to assume a single input value is used.



Steps Involved in Supervised Learning:

- First Determine the type of training dataset.
- Collect/Gather the labelled training data.
- Split the training dataset into training dataset, test dataset, and validation dataset.
- Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

1. Linear Regression
2. Regression Trees
3. Non-Linear Regression
4. Bayesian Linear Regression
5. Polynomial

Regression Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

- Spam Filtering,
- Random Forest
- Decision Trees
- Logistic Regression
- Support vector Machines

CHAPTER 2

LITERATURE

SURVEY

Bus Pooling: A Large-Scale Bus Ridesharing Service

Slugging is a variation of ride-share commuting and hitchhiking. The slugging problem was proposed in [19]. It assumes that passenger A abandons her trip and joins B's trip (i.e., A's trip is merged into B's trip) and considers a vehicle's capacity constraint to be two to five passengers. From the procedure, slugging is very similar to our proposed model if only one pickup/delivery point is considered in a practical setting. But in fact, a better pickup/delivery point for passengers in our proposed model is designed.

Rather than B waiting for A, it is more reasonable to pick the location nearest to both A and B. In addition, we assume a homogeneous fleet of vehicles with a 30-seat capacity (or even more) because our goal is to solve the problem of high capacity.

Each driver $d = hCapacity, Threshold, EarliestTime, LatestTime_i$ has two static values: Capacity for the number of seats on a bus and Threshold for the number of riders that meet the departure threshold. Each driver has a given time-window when riders are required to arrive; if they are early, they have to wait and if they are late, the driver can refuse to pick them up. The lower bound of the time-window, EarliestTime, denotes the time when the driver arrives at the pickup point, which is the earliest start time for the service. The upper bound of the timewindow, LatestTime, denotes the departure time when the driver leaves the pickup point, which is the latest start time for the service.

URoad: An Efficient Algorithm for Large-scale Dynamic Ridesharing Service

Unlike taxis, the driver V who provides ridesharing service has his own schedule. Therefore, besides his origin, $V.o$, and destination, $V.d$, he will be allowed to

submit the constraints as follows: the

departure time, V_{dt} , when he will start from V_o , the latest arrival time, V_{at} , when he must arrive at V_d before and the number of available seats, V_{as} , that he can provide for ridesharing. Passengers. Similar to the driver, the passenger R also has his own schedule. So, besides his origins, R_o , and destination, R_d , his ridesharing request will contain the departure time period, which includes the earliest time, $R_{dtp.e}$, and the latest time, $R_{dtp.l}$, for driver to arrive at R_o and pick up him, the latest arrival time, R_{at} , when he must arrive at R_d before and the maximum price, R_{mp} , which he is willing to pay for this ridesharing service. When URoad finds the optimal driver, R will receive the driver's information and just wait for the driver to pick him up during departure time period.

The cost for the original trip of R_i ; 2) DV_{i,R_i} , the detour cost caused by sharing the ride with R_i ; 3) BR_{i,R_j} , the discount fare when a new passenger R_j joins in after the passenger R_i . The motivation behind BR_{i,R_j} is that the whole trip distance increases due to the new passenger whose destination as well as his constraints may result in delaying for the existing passengers, hence former passengers need to get compensations.

RideMatcher: Peer-to-peer Matching of Passengers for Efficient Ridesharing

It describes the components of the RideMatcher system and the interactions between them. The main feature of our system is decentralization. In RideMatcher, all mobile nodes (e.g., personal cars, taxis, pedestrians) perform only local operations, without relying on a central coordinator. The goal of each mobile node is to group with other mobile nodes that have similar routes at a certain time, such that sharing a car ride with these nodes cuts down the cost of the trip.

In pursuing this goal, each mobile node is selfish, being interested only in obtaining a higher benefit for itself, without caring about other nodes in the system. However, in order to ensure that the system converges to a stable state, we define a set of rules in order to prevent situations when nodes keep grouping and

ungrouping indefinitely.

Our system is based on our earlier PeerMatcher system [10]. However, as stated in section II, there are several changes that have to be made in order to adapt PeerMatcher to the ridesharing problem.

PhD Forum: Toward a Human-in-the-Loop Smart Ridesharing System with Self-Driving Technologies

To enable stable routing, one way to do that is to utilize a probabilistic road network model for route computation. In such a model, the goal is to maximize the probability of a path at arriving its destination before an arbitrary time deadline by considering travel time, risk and deadline simultaneously [6]. However, in order to adopt such model, we must consider the impact of combining it with dynamic road conditions. It is because the traffic impact brings forward by dynamic road conditions is short-term; the impact will last only for some duration. One approach would be to merge the batch updates with the probabilistic road model, which will cause the model to recompute itself based on the intensity of change. It is also possible to continuously learn and refine the probabilistic road network model in an online fashion using the dynamic road condition streams. Another approach would be to do routing from a combination of the probabilistic road model and a periodically updated road network.

This protocol is called upon when the original vehicle could no longer satisfy his or her time constraint due to the delay caused by current road conditions and because of the schedule constraint imposed by other onboard passengers.

Similarly, the incentive protocol also relies on the result from the fast affected-passenger discovery to compute and compromise compensation for the affected passengers. The goal of the incentive protocol is to (i) avoid the physical inconvenience from transferring and to

(ii) increase user satisfaction on ridesharing services even with the presence of unexpected road conditions, thus encouraging future use of service.

CHAPTER 3

SYSTEM

ANALYSIS

One of the primary choices we face for our project implementations is “Which development methodology should we use?” There are various software development methodologies available in the ICT industry. They are

- Agile Software Development
- Crystal Methods
- Dynamic Systems Development Model (DSDM)
- Extreme Programming (XP)
- Feature Driven Development (FDD)
- Joint Application Development (JAD)
- Lean Development (LD)
- Rapid Application Development (RAD)
- Rational Unified Process (RUP)
- Scrum
- Spiral
- Systems Development Life Cycle (SDLC)
- Waterfall (a.k.a. Traditional)

Among all these methodologies, we have decided to compare the two most popular and widely used methodologies:

- Waterfall Model
- RAD Model

4.2 Waterfall Model

It is called as traditional approach. Waterfall is a linear method (sequential model) for any software application development. Here application development is segregated into a sequence of pre-defined phases.

Waterfall Model (Taken from Google.com)

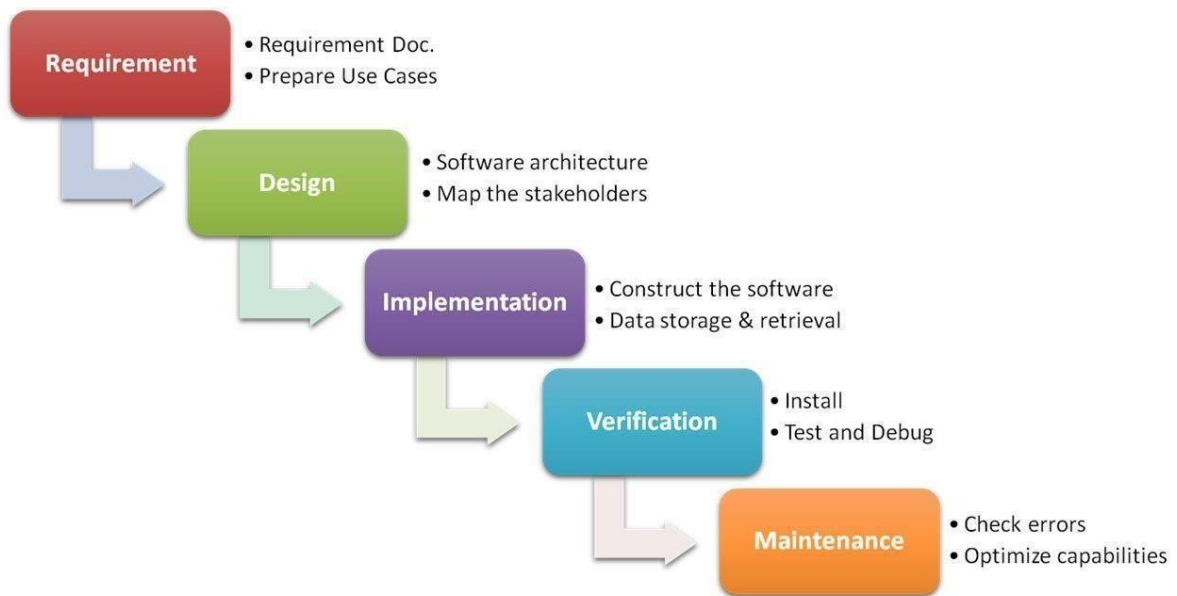


Fig 3.1

(Reference:<https://www.lucidchart.com/blog/waterfall-project-management-methodology>)

1. Requirement gathering and documentation

In this stage, you should gather comprehensive data approximately what this challenge requires. You may gather this data in a diffusion of ways, from interviews to questionnaires to interactive brainstorming. By means of the stop of this section, the task requirements have to be clean, and also you have to have a necessities file that has been allotted on your team.

2. System design

Using the well-known requirements, your team designs the solutions. During this phase, no development will be happening. But the project team starts specification such as programming language or hardware requirements.

3. Implementation

During this phase software development coding will be happening. Web application programmers take data from the previous stage and create a functional product. Web application programmers write source code in small pieces, which

are integrated at the end of this phase or the beginning of the next.

4. Testing

Once all coding is done, testing of the product can begin. Testers methodically find and report any problems. If serious issues arise, your project may need to return to phase one for reevaluation.

5. Delivery/deployment

In this phase, the solution is complete, and your project team submits the deliverables to be deployed or released.

6. Maintenance

The final solution has been implemented to the client and is being used. As troubles arises, the project team might also want to create patches and updates might also to deal with them. Again, huge troubles may also necessitate a return to segment one.

4.3 RAD Model

Agile is an iterative, team-based approach to development. This method highlights the speedy delivery of an software application in comprehensive functional components.

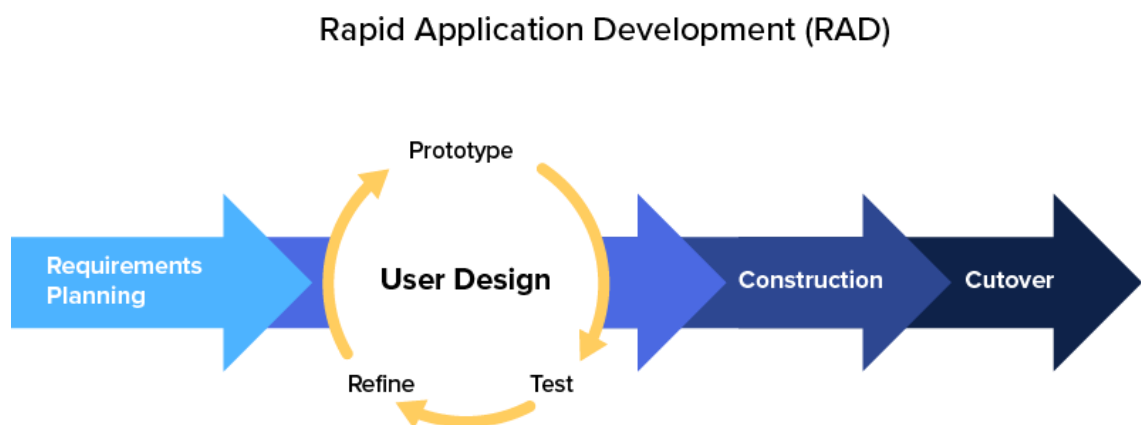


Fig 3.2

RAD Model (Taken from Google.com)

References:(<https://www.lucidchart.com/blog/rapid-application-development-methodology>)

Phase 1: Requirements planning

This phase is equivalent to a project scoping meeting. Although the planning phase is condensed compared to other project management methodologies, this is a critical step for the ultimate success of the project.

During this stage, web application programmers, clients (software users), and team members communicate to determine the goals and expectations for the project as well as current and potential issues that would need to be addressed during the build.

A basic breakdown of this stage involves:

Researching the current problem

Defining the requirements for the
project

Finalizing the requirements with each stakeholder's approval

It is important that everyone has the opportunity to evaluate the goals and expectations for the project and weigh in. By getting approval from each key stakeholder and web application programmer, teams can avoid miscommunications and costly change orders down the road.

Phase: User design

Once the project is scoped out, it's time to jump right into development, building out the user design through various prototype iterations.

This is the meat and potatoes of the RAD methodology—and what sets it apart from other project management strategies. During this phase, clients work hand in

hand with web application programmers to ensure their needs are being met at every step in the design process. It's almost like customizable software development where the users can test each prototype of the product, at each stage, to ensure it meets their expectations.

All the bugs and kinks are worked out in an iterative process. The web application programmer designs a prototype, the client (user) tests it, and then they come together to communicate on what worked and what didn't.

This method gives web application programmers the opportunity to tweak the model as they go until they reach a satisfactory design.

Both the software web application programmers and the clients learn from the experience to make sure there is no potential for something to slip through the cracks.

Phase 3: Rapid construction

Phase 3 takes the prototypes and beta systems from the design phase and converts them into the working model.

Because the majority of the problems and changes were addressed during the thorough iterative design phase, web application programmers can construct the final working model more quickly than they could by following a traditional project management approach.

The phase breaks down into several smaller steps:

- 1.Preparation for rapid construction
- 2.Program and application development
- 3.Coding Unit, integration, and system

testing

The software development team of programmers, coders, testers, and web application programmers work together during this stage to make sure everything is working smoothly and that the end result satisfies the client's expectations and

objectives.

This third phase is important because the client still gets to give input throughout the process. They can suggest alterations, changes, or even new ideas that can solve problems as they arise.

Phase 4: Cutover

This is the implementation phase where the finished product goes to launch. It includes data conversion, testing, and changeover to the new system, as well as user training.

All final changes are made while the coders and clients continue to look for bugs in the system.

Benefits of RAD methodology

RAD is one of the most successful software development programs available today, with numerous benefits for both software development teams as well as their clients.

Here are just a few advantages:

RAD lets you break the project down into smaller, more manageable tasks.

The task-oriented structure allows project managers to optimize their team's efficiency by assigning tasks according to members' specialties and experience.

Clients get a working product delivered in a shorter time frame.

Regular communication and constant feedback between team members and stakeholders increases the efficiency of the design and build process.

3.1 EXISTING SYSTEM

The existing application of ridesharing strategy to autonomous taxi system holds great promise for improving the efficiency in the future on-demand ride-hailing services. Prior to the implementation of dispatching strategies to the autonomous taxi system, it is necessary to gain insight into the performance of the

dispatching strategies. This study aims to solve the dynamic ridesharing problem and conduct a comprehensive quantity analysis of the ridesharing efficiency for various demand levels and carrying capacities in a metropolitan area. The framework involves three core modules: Searching, Scheduling and Waiting. All of the active requests in very short period will be sorted by a scheme first-in-first-out. And server constantly updates the vehicles' Riding Fares and status. In each calculation period, the searching module retrieves the appropriate taxis with the constraints for the requests. After that, the scheduling module matches the orders with the selected taxis and compares all of the possible paths. Finally, the system updates the route and schedule of the best-matched vehicle. The order will be put into the waiting module if any vehicles are failed to match during the above process.

3.2 DRAWBACKS OF EXISITING SYSTEM

- Prone to Errors

- Can't learn relation between factors.

- Do not take account of the influence features.

- Maximizes the complexity of the problem.

3.3 PROPOSED SYSTEM

The proposed system assumes that most passengers are willing to take a taxi ridesharing, which coincides with a recent report [23]. Specifically, the passengers either explicitly report their ride requests to the system through booking Apps or implicitly join the ridesharing by hailing a shared taxi at roadside. The system will select proper taxis to serve both online and offline ride requests by analysing current statuses of all taxis, and plan routes for the chosen taxis given certain service requirements.

The proposed system has mentioned two main processes: request handling plus assignment, and the remediation protocols for handling unexpected road conditions. Both components have one common issue: neither of them has

considered the factor of route stability in the underlying road network with respect to dynamic road conditions. In the context of this work, route stability refers to the number of times the need of rerouting or transferring appears for a ridesharing vehicle driver. In other words, we could also view it as the probability that some unexpected road conditions are going to affect the time deadlines of a set of trip schedule given a particular trip schedule assignment (which consists of routes for the assigned users) of a driver.

The proposed scheme searches candidate taxis with only the origin Riding Fare of a ride request. Even though both origin and destination are utilized for a dual-side search, they are separately considered in these works. Besides, previous works will immediately return once a valid taxi is found, rather than searching for the best one. Such a partial trip information based passenger-taxi matching can not filter out the invalid taxis at the beginning, while it meanwhile may also miss the best taxi that introduces the minimum cost.

3.4 ADVANTAGES OF PROPOSED SYSTEM

- Efficient passenger-taxi matching

- Supports Longitude and latitude-based ridesharing.

- Requests are dynamically updated.

- Outstanding Learning Capabilities

- Effectively improve prediction accuracy

CHAPTER 4

REQUIREMENT

SPECIFICATION

4.1 INTRODUCTION

Clearly defined requirements are essential signs on the road that leads to a successful project. They establish a formal agreement between a client and a provider that they are both working to reach the same goal. High-quality, detailed requirements also help mitigate financial risks and keep the project on a schedule. According to the Business Analysis Body of Knowledge definition, requirements are a usable representation of a need.

Creating requirements is a complex task as it includes a set of processes such as elicitation, analysis, specification, validation, and management. In this article, we'll discuss the main types of requirements for software products and provide a number of recommendations for their use.

4.2 HARDWARE SPECIFICATION

4.2.1 HARDWARE REQUIREMENT

Hardware	Minimum Requirement
Disk Space	32 GB or more,10 GB or more for Foundation Edition
Processor	1.4 GHz 64 bit
Memory	512 MB
Display	(800 × 600) Capable video adapter and monitor

4.2.2 SOFTWARE SPECIFICATION

Backend Technologies

- ✓ Python
- ✓ Numpy
- ✓ Sci-learn
- ✓ Eclipse

IDE Frontend

Technologies

- ✓ Web Technologies
- ✓ Bootstrap

4.3 SOFTWARE DESCRIPTION

4.3.1 ABOUT PYTHON

Python is a free, open-source programming language. Therefore, all you have to do is install Python once, and you can start working with it. Not to mention that you can contribute your own code to the community. Python is also a cross-platform compatible language. So, what does this mean? Well, you can install and run Python on several operating systems. Whether you have a Windows, Mac or Linux, you can rest assure that Python will work on all these operating systems.

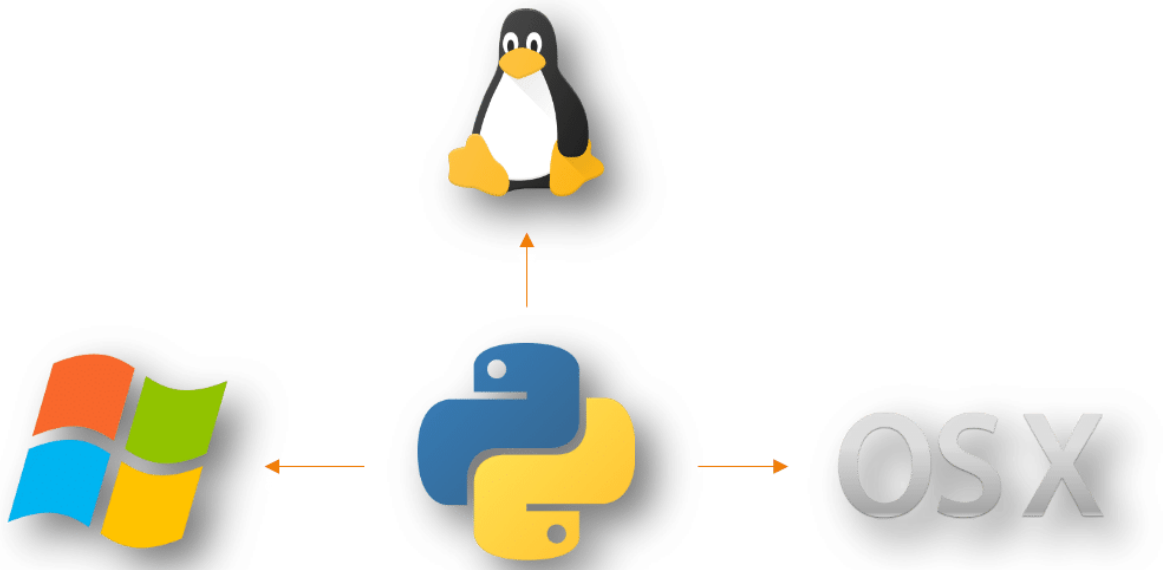


Fig 4.1

Python is also a great visualization tool. It provides libraries such as Matplotlib, seaborn and bokeh to create stunning visualizations.



Fig 4.2

In addition, Python is the most popular language for machine learning and deep learning. As a matter of fact, today, all top organizations are investing in Python to implement machine learning in the back-end.

PYTHON LINE STRUCTURE

Python coding style comprises physical lines as well as logical lines or

statements. A physical line in a Python program is a sequence of characters, and the end of the line terminates the line sequence as opposed to some other languages, such as C and C++ where a semi-colon is used to mark the end of the statement. A logical line, on the other hand, is composed of one or more physical lines. The use of a semi-colon is not prohibited in Python, although it's not mandatory. The NEWLINE token denotes the end of the logical line. A logical line that only contains spaces, comments, or tabs are called blank lines and they are ignored by the interpreter.

As we saw that in Python, a new line simply means that a new statement has started. Although, Python does provide a way to split a statement into a multiline statement or to join multiple statements into one logical line. This can be helpful to increase the readability of the statement. Following are the two ways to split a line into two or more lines:

Explicit Line Joining :

In explicit line joining, we use a backward slash to split a statement into a multiline statement.

Implicit Line Joining :

Statements that reside inside [], {}, or () parentheses can be broken down into two or more physical lines without using a back slash.

Multiple Statements on a Single Line :

In Python, it is possible to club multiple statements in the same line using a semi-colon; however, most programmers do not consider this to be a good practice as it reduces the readability of the code.

Whitespaces and Indentation :

Unlike most of the programming languages, Python uses indentation to mark a block of code. According to Python coding style guideline or PEP8, we should keep an indent size of four.

Most of the programming languages provide indentation for better code formatting

and don't enforce to have it. But in Python it is mandatory. This is why indentation is so crucial in Python.

Comments in any programming language are used to increase the readability of the code. Similarly, in Python, when the program starts getting complicated, one of the best ways to maintain the readability of the code is to use Python comments. It is considered a good practice to include documentations and notes in the python syntax since it makes the code way more readable and understandable to other programmers as well, which comes in handy when multiple programmers are simultaneously working on the same project.

The code can only explain how it does something and not why it does that, but Python comments can do that. With Python comments, we can make documentations for various explanations in our code itself. Comments are nothing but tagged lines of codes which increase the readability of a code and make it self-explanatory. There are different ways of creating comments depending on the type of comment we want to include in our code.

Following are different kinds of comments that can be included in our Python program:

1. Single Line Comments
2. Docstring Comments
3. Multiline Comments

Single line Python comments are marked with `#` character. These comments end at the end of the physical line, which means that all characters starting after the `#` character (and lasts till the end of the line) are part of the comment.

Python has the documentation strings (or docstrings) feature which is usually the first statement included in functions and modules. Rather than being ignored by the Python

Interpreter like regular comments, docstrings can actually be accessed at the run time using the dot operator.

It gives programmers an easy way of adding quick notes with every Python module, function, class, and method. To use this feature, we use triple quotes in the beginning of the documentation string or comment and the closing triple quotes at the end of the documentation comment. Docstrings can be one-liners as well as multi-liners.

Unlike some programming languages that support multiline comments, such as C, Java, and more, there is no specific feature for multiline comments in Python. But that does not mean that it is totally impossible to make multiline comments in Python. There are two ways we can include comments that can span across multiple lines in our Python code.

Python Block Comments: We can use several single line comments for a whole block. This type of comment is usually created to explain the block of code that follows the Block comment. Python Block comment is the only way of writing a real comment that can span across multiple lines. It is supported and preferred by Python's PEP8 style guide since Block comments are ignored by Python interpreter or parser. However, nothing is stopping programmers from using the second 'non-real' way of writing multiline comments in Python which is explained below.

Using Docstrings: Docstrings are largely used as multiline comments in Python by many programmers since it is the closest thing to having a multiline comment feature in Python. While it is not wrong to use docstrings when we need to make multiline comments, it is important to keep in mind that there is a significant difference between docstrings and comments. Comments in Python are totally ignored by the Python Interpreter, while docstrings, when used inside the Python function, can be accessed at the run time.

Python Data Types

One of the most crucial part of learning any programming language is to

understand how data is stored and manipulated in that language. Users are often inclined toward Python because of its ease of use and the number of versatile features it provides. One of those features is dynamic typing.

In Python, unlike statically typed languages like C or Java, there is no need to specifically declare the data type of the variable. In dynamically typed languages such as Python, the interpreter itself predicts the data type of the Python Variable based on the type of value assigned to that variable.

Advantages of Python

- Universal Language Construct
- Support both High Level and Low Level Programming
- Language Interoperability
- Fastest Development life cycle therefore more productive coding environment
- Less memory used because a single container hold multiple data types and each type doesn't require its own function
- Learning Ease and open source development
- Speed and user-friendly data structure
- Extensive and extensible libraries.
- Simple & support IoT
- And many more

Based on the 3 languages that are described above, we decided to use Python as programming language for developing this e-Voting web based application. The important motives are

- Easy to learn, even non experienced programmers can use it. Ex: spacing and tabbing instead of extra syntax
- Interactive mode
- Large and comprehensive standard libraries
- Python programs resemble to that of pseudo-code. This makes it a basis and a must have for beginner programmers due to its extreme ease a

difficulty when compared to C++ Java, Perl, and so forth

About Pandas

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures.

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn.

Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily.

Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to visualize pandas' DataFrames and plots. As a matter of fact, this article was created entirely in a Jupyter Notebook.

There are two types of data structures in pandas: Series and DataFrames.

1. Series: a pandas Series is a one dimensional data structure (“a one dimensional ndarray”) that can store values — and for every value it holds a unique index, too.

2. DataFrame: a pandas DataFrame is a two (or more) dimensional data structure – basically a table with rows and columns. The columns have names and the rows have indexes.

Those who are familiar with R know the data frame as a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a data frame's rows do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc.

Now, DataFrames in Python are very similar: they come with the Pandas library, and they are defined as two-dimensional labeled data structures with columns of potentially different types.

In general, you could say that the Pandas DataFrame consists of three main components: the data, the index, and the columns.

Firstly, the DataFrame can contain data that is:

- a Pandas DataFrame.
- a Pandas Series: a one-dimensional labeled array capable of holding any data type with axis labels or index. An example of a Series object is one column from a DataFrame.
- a NumPy ndarray, which can be a record or structured.
- a two-dimensional ndarray.
- dictionaries of one-dimensional ndarray's, lists, dictionaries or Series.

Some of the key features of Python Pandas are as follows:

It provides DataFrame objects with default and customized indexing which is very fast and efficient.

- There are tools available for loading data of different file formats into

in-memory data objects.

- It is easy to perform data alignment and integrated handling of missing data in Python Pandas..
- It is very simple to perform pivoting and reshaping of data sets in Pandas.
- We can easily insert and delete columns from a data structure.
- Data aggregation and transformations can be done using group by..
- High-performance merging and joining of data can be done using Pandas.
- It also provides time series functionality.
- Inserting and deleting columns in data structures.
- Merging and joining data sets.
- Reshaping and pivoting data sets.
- Aligning data and dealing with missing data.
- Manipulating data using integrated indexing for DataFrame objects.
- Performing split-apply-combine on data sets using the group by engine.
- Manipulating high-dimensional data in a data structure with a lower dimension using hierarchical axis indexing.
- Subsetting, fancy indexing, and label-based slicing data sets that are large in size.
- Generating data range, converting frequency, date shifting, lagging, and other time-series functionality.
- Reading from files with CSV, XLSX, TXT, among other formats.
- Arranging data in an order ascending or descending.
- Filtering data around a condition.
- Analyzing time series.
- Iterating over a data set.

Numpy

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy.

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays.

An array is basically nothing but pointers. It's a combination of a memory address, a data type, a shape, and strides:

- The data pointer indicates the memory address of the first byte in the array,
- The data type or dtype pointer describes the kind of elements that are contained within the array,
- The shape indicates the shape of the array, and
- The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next row.

Or, in other words, an array contains information about the raw data, how to locate an element and how to interpret an element.

With NumPy, we work with multidimensional arrays. We'll dive into all of the possible types of multidimensional arrays later on, but for now, we'll focus on 2-dimensional

arrays. A 2-dimensional array is also known as a matrix, and is something you should be familiar with. In fact, it's just a different way of thinking about a list of lists. A matrix has rows and columns. By specifying a row number and a column number, we're able to extract an element from a matrix.

We can create a NumPy array using the `numpy.array` function. If we pass in a list of lists, it will automatically create a NumPy array with the same number of rows and columns. Because we want all of the elements in the array to be float elements for easy computation, we'll leave off the header row, which contains strings. One of the limitations of NumPy is that all the elements in an array have to be of the same type, so if we include the header row, all the elements in the array will be read in as strings. Because we want to be able to do computations like find the average quality of the wines, we need the elements to all be floats.

NumPy has several advantages over using core Python mathematical functions, a few of which are outlined here:

1. NumPy is extremely fast when compared to core Python thanks to its heavy use of C extensions.
2. Many advanced Python libraries, such as Scikit-Learn, Scipy, and Keras, make extensive use of the NumPy library. Therefore, if you plan to pursue a career in data science or machine learning, NumPy is a very good tool to master.

NumPy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

Mathplotlib

Plotting of data can be extensively made possible in an interactive way by

Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib.

Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib, and its lead developer was Michael Droettboom. One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams.

Matplotlib is a plotting library like GNUplot. The main advantage towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

Another reason for the attractiveness of Matplotlib lies in the fact that it is widely considered to be a perfect alternative to MATLAB, if it is used in combination with Numpy and Scipy. Whereas MATLAB is expensive and closed source, Matplotlib is free and open source code. It is also object-oriented and can be used in an object oriented way. Furthermore it can be used with general-purpose GUI toolkits like wxPython, Qt, and GTK+. There is also a procedural "pylab", which designed to closely resemble that of MATLAB. This can make it extremely easy for MATLAB users to migrate to matplotlib.

Matplotlib can be used to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Another characteristic of matplotlib is its steep learning curve, which means that users usually make rapid progress after having started. The official website has to say the following about this: "matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code."

4.3.2 ABOUT IDE

IDE – Integrated Development Environment. It is software suite. It contains different instruments for web application software engineers to compose and test applications.

Programming Web application software engineers utilize wide scope of devices habitually all through application source code creation, building and application testing. Programming Development apparatuses frequently incorporate word processors for source code creation, code libraries for reuse, compilers and test stages.

An Integrated Development Environment (IDE) is a product application that gives a programming domain to streamline creating and troubleshooting programming. Instead of playing out every one of the means required to make an executable program as disconnected individual assignments, it brings every one of the devices required into one application and workspace. Every one of the devices has a familiarity with the earth, and they cooperate to display a consistent improvement set for the designer.

An IDE typically has an editor and a compiler window. There, we can write our code, run it, and compile it. Hence, it becomes much easier to develop applications using an IDE. The major reasons for using an IDE are given below:

- We can easily write our code in its text editor window.

- To store the resource files of a project, we have a project editor window.

- It makes debugging of the code easier.

- An IDE's autosuggest feature helps us in writing efficient code.

- We can import other projects for integrating with the existing one.

Without an IDE, a product web application software engineer must choose, send, incorporate and deal with these instruments independently. An IDE brings huge numbers of those improvement related instruments all together system, application or administration. The incorporated toolset is intended to streamline programming advancement and can distinguish and limit coding missteps and grammatical

errors.

- ✓ Komodo
- ✓ NetBeans
- ✓ PyCharm
- ✓ Python for Visual Studio.NET
- ✓ PyDev
- ✓ PyStudio
- ✓ LiClipse(Eclipse)

LiClipse is at the core for all Python based application development activities. It's been available for many years. LiClipse is a set of plugins to enhance Eclipse. It improves the overall Eclipse experience. It is best IDE tool to develop Python Applications.

Advantages of LiClipse

- Perfect Debugger
- Error checking features.
- An enhanced ability to analyze the code for quality and security concerns.
- Auto complete feature

Based on the various IDE that are described above, we decided to use LiClipse as IDE (Integrated Development Environment) for developing this e-Voting web based application.

Important Features That Make LiClipse our IDE of Choice

- Rich Reusable Libraries are available
- Model Driven Development
- Enterprise Python Tooling
- Powerful Debugging options

IDE is a program that is used to write applications (other programs) in a variety of programming languages. It allows you to complete many distinct program development steps in one place (program window). A program which allows you to

do this is known as an integrated development environment, or IDE. The program is free and easy to import onto your home computer.

LiClipse is used for writing programs in Python. There are several steps that must be completed to set up LiClipse and further steps to start up a new Python project (set of Python components which work together to make up an application).

PyCharm is an Integrated Development Environment (IDE) that provides tools for writing efficient code. It supports various frameworks such as Django, Flask, and Pyramid. PyCharm is majorly used by developers for application development with Python language.

Let's suppose, we are making a project using Python programming language. We know that Python is extensively popular among the developer's community, as well as in industries, due to its following features:

Simple to code: Python is a very simple language compared to other programming languages. Also, it is a developer-friendly language.

Object-oriented language: Python is an object-oriented language that uses concepts of classes and objects.

GUI programming support: We can create graphical user interfaces using Python Tkinter library. Besides, we can use modules such as PyQt5, PyQt4, and wxPython to create a GUI application.

High-level language: Python is a high-level programming language. We do not need to take care of managing memory or the system architecture while writing Python codes.

Portability: Python is a portable programming language, i.e., we can run Python codes on any platform. For example, we can run a code written for Windows in Linux, UNIX, and Mac, without changing it.

Integrated language: Python is an integrated programming language. We can easily integrate C, C++, etc. with Python.

Interpreted language: Python is an interpreted programming language as well. Python's source code is changed to bytecode. Due to this, it is directly executed without compiling.

Large set of standard libraries: Python consists of a rich set of modules, packages, and libraries. It helps in various stages of application development.

"notebook" or "notebook documents" denote documents that contain both code and rich text elements, such as figures, links, equations, ... Because of the mix of code and text elements, these documents are the ideal place to bring together an analysis

description, and its results, as well as, they can be executed perform the data analysis in real time.

The Jupyter Notebook App produces these documents.

As a server-client application, the Jupyter Notebook App allows you to edit and run your notebooks via a web browser. The application can be executed on a PC without Internet access, or it can be installed on a remote server, where you can access it through the Internet.

Its two main components are the kernels and a dashboard.

A kernel is a program that runs and introspects the user's code. The Jupyter Notebook App has a kernel for Python code, but there are also kernels available for other programming languages.

The dashboard of the application not only shows you the notebook documents that you have made and can reopen but can also be used to manage the kernels: you can which ones are running and shut them down if necessary.

CHAPTER 5

SYSTEM

DESIGN

5.1 INTRODUCTION

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

Mainly, UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into the documentation of several business processes or workflows. For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts. They provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficiency.

5.2 ARCHITECTURE DIAGRAM

An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components.

What is a diagram? What are the types of diagrams for architecture? The Dragon1 open EA Method makes it very clear: if a diagram does not show a concept, principle or part of a principle, it is NOT an architecture diagram, because it does not show (a part of) the architecture.

There are many kinds of architecture diagrams, like a software architecture diagram, system architecture diagram, application architecture diagram, security architecture diagram, etc.

For system developers, they need system architecture diagrams to understand, clarify, and communicate ideas about the system structure and the user requirements that the system must support. It's a basic framework can be used at

the system planning phase

helping partners understand the architecture, discuss changes, and communicate intentions clearly.

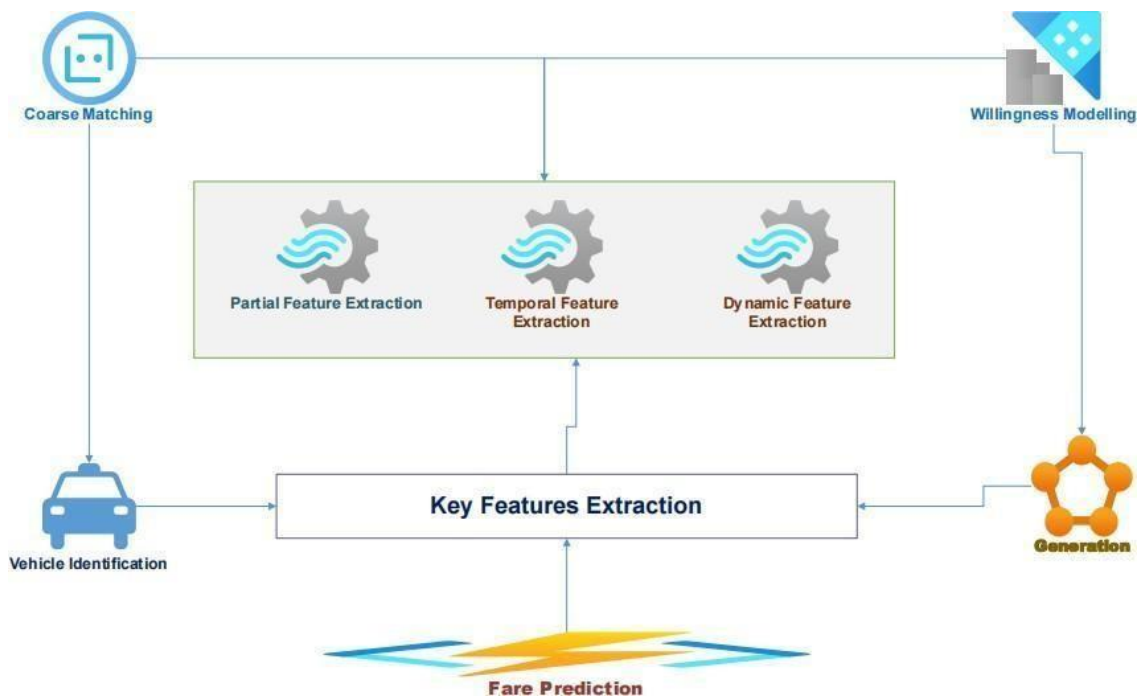


Fig 5.1

CHAPTER 6

SYSTEM

IMPLEMENTATION

6.1 List of Modules

6.1.1 EXPLORATORY DATA EVALUATION

Exploratory Data Analysis (EDA) is the first step in your data analysis process. Here, you make sense of the data you have and then figure out what questions you want to ask and how to frame them, as well as how best to manipulate your available data sources to get the answers you need. You do this by taking a broad look at patterns, trends, outliers, unexpected results and so on in your existing data, using visual and quantitative methods to get a sense of the story this tells.

Exploratory Data Analysis is valuable to data science projects since it allows to get closer to the certainty that the future results will be valid, correctly interpreted, and applicable to the desired business contexts. Such level of certainty can be achieved only after raw data is validated and checked for anomalies, ensuring that the data set was collected without errors. EDA also helps to find insights that were not evident or worth investigating to business stakeholders and data scientists but can be very informative about a particular business.

Stakeholders and data scientists but can be very informative about a particular business.

EDA is performed in order to define and refine the selection of feature variables that will be used for machine learning. Once data scientists become familiar with the data set, they often have to return to feature engineering step, since the initial features may turn out not to be serving their intended purpose. Once the EDA stage is complete, data scientists get a firm feature set they need for supervised and unsupervised machine learning.

6.1.2 PRE-PROCESSING

Sometimes you may find some data are missing in the dataset. We need to be equipped to handle the problem when we come across them. Obviously you could remove the entire line of data but what if you are unknowingly removing crucial information? Of course we would not want to do that. One of the most common idea to handle the problem is to take a mean of all the values of the same column and have it to replace the missing data.

The library that we are going to use for the task is called Scikit Learn preprocessing. It contains a class called Imputer which will help us take care of the missing data.

Sometimes our data is in qualitative form, that is we have texts as our data. We can find categories in text form. Now it gets complicated for machines to understand texts and process them, rather than numbers, since the models are based on mathematical equations and calculations. Therefore, we have to encode the categorical data.

Now we need to split our dataset into two sets — a Training set and a Test set. We will train our machine learning models on our training set, i.e our machine learning models will try to understand any correlations in our training set and then we will test the models on our test set to check how accurately it can predict. A general rule of the thumb is to allocate 80% of the dataset to training set and the remaining 20% to test set. For this task, we will import `test_train_split` from `model_selection` library of `scikit`.

6.1.3 FEATURE ENGINEERING

Filter methods are generally used as a preprocessing step. The selection of features is independent of any machine learning algorithms. Instead, features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable. The correlation is a subjective term here. For basic guidance, you can refer to the following table for defining correlation co-efficients.

Pearson's Correlation: It is used as a measure for quantifying linear dependence between two continuous variables X and Y. Its value varies from -1 to +1.

LDA: Linear discriminant analysis is used to find a linear combination of features that characterizes or separates two or more classes (or levels) of a categorical variable.

ANOVA: ANOVA stands for Analysis of variance. It is similar to LDA except for the fact that it is operated using one or more categorical independent features and one continuous dependent feature. It provides a statistical test of whether the means of several groups are equal or not.

Chi-Square: It is a statistical test applied to the groups of categorical features to evaluate the likelihood of correlation or association between them using their frequency distribution.

6.1.4 PREDICTION

Once training is complete, it's time to see if the model is any good, using Evaluation. This is where that dataset that we set aside earlier comes into play. Evaluation allows us to test our model against data that has never been used for training. This metric allows us to see how the model might perform against data that it has not yet seen. This is meant to be representative of how the model might perform in the real world.

A good rule of thumb I use for a training-evaluation split somewhere on the order of 80/20 or 70/30. Much of this depends on the size of the original source dataset. If you have a lot of data, perhaps you don't need as big of a fraction for the evaluation dataset.

Once you've done evaluation, it's possible that you want to see if you can further improve your training in any way. We can do this by tuning our parameters. There were a few parameters we implicitly assumed when we did our training, and now is a good time to go back and test those assumptions and try other values.

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions

and decision making. As the name goes, it uses a tree-like model of decisions. A decision tree is drawn upside down with its root at the top. In the image on the left, the bold text in black represents a condition/internal node, based on which the tree splits into branches/ edges. The end of the branch that doesn't split anymore is the decision/leaf, in this case, whether the passenger died or survived, represented as red and green text respectively.

Although, a real dataset will have a lot more features and this will just be a branch in a much bigger tree, but you can't ignore the simplicity of this algorithm. The feature importance is clear and relations can be viewed easily. This methodology is more commonly known as learning decision tree from data and above tree is called Classification tree as the target is to classify passenger as survived or died. Regression trees are represented in the same manner, just they predict continuous values like price of a house. In general, Decision Tree algorithms are referred to as CART or Classification and Regression Trees.

CHAPTER 7

SOFTWARE

TESTING

Testing documentation is the documentation of artifacts that are created during or before the testing of a software application. Documentation reflects the importance of processes for the customer, individual and organization. Projects which contain all documents have a high level of maturity. Careful documentation can save the time, efforts and wealth of the organization.

If the testing or development team gets software that is not working correctly and developed by someone else, so to find the error, the team will first need a document. Now, if the documents are available then the team will quickly find out the cause of the error by examining documentation. But, if the documents are not available then the tester need to do black box and white box testing again, which will waste the time and money of the organization. More than that, Lack of documentation becomes a problem for acceptance.

Benefits of using Documentation

- Documentation clarifies the quality of methods and objectives.

- It ensures internal coordination when a customer uses software application.

- It ensures clarity about the stability of tasks and performance.

- It provides feedback on preventive tasks.

- It provides feedback for your planning cycle.

- It creates objective evidence for the performance of the quality management system.

The test scenario is a detailed document of test cases that cover end to end functionality of a software application in liner statements. The liner statement is considered as a scenario. The test scenario is a high-level classification of testable requirements. These requirements are grouped on the basis of the functionality of a module and obtained from the use cases.

In the test scenario, there is a detailed testing process due to many associated test cases. Before performing the test scenario, the tester has to consider the test cases for each scenario.

In the test scenario, testers need to put themselves in the place of the user because they test the software application under the user's point of view. Preparation of scenarios is the most critical part, and it is necessary to seek advice or help from customers, stakeholders or developers to prepare the scenario.

As per the IEEE Documentation describing plans for, or results of, the testing of a system or component, Types include test case specification, test incident report, test log, test plan, test procedure, test report. Hence the testing of all the above mentioned documents is known as documentation testing.

This is one of the most cost effective approaches to testing. If the documentation is not right: there will be major and costly problems. The documentation can be tested in a number of different ways to many different degrees of complexity. These range from running the documents through a spelling and grammar checking device, to manually reviewing the documentation to remove any ambiguity or inconsistency.

Documentation testing can start at the very beginning of the software process and hence save large amounts of money, since the earlier a defect is found the less it will cost to be fixed.

The most popular testing documentation files are test reports, plans, and checklists. These documents are used to outline the team's workload and keep track of the process. Let's take a look at the key requirements for these files and see how they contribute to the process.

Test strategy. An outline of the full approach to product testing. As the project moves along, developers, designers, product owners can come back to the document and see if the actual performance corresponds to the planned activities.

Test data. The data that testers enter into the software to verify certain features and their outputs. Examples of such data can be fake user profiles, statistics, media content, similar to files that would be uploaded by an end-user in a ready solution.

Test plans. A file that describes the strategy, resources, environment, limitations, and schedule of the testing process. It's the fullest testing document, essential for informed planning. Such a document is distributed between team members and shared with all stakeholders.

Test scenarios. In scenarios, testers break down the product's functionality and interface by modules and provide real-time status updates at all testing stages. A module can be described by a single statement, or require hundreds of statuses, depending on its size and scope.

Test cases. If the test scenario describes the object of testing (what), a scenario describes a procedure (how). These files cover step-by-step guidance, detailed conditions, and current inputs of a testing task. Test cases have their own kinds that depend on the type of testing — functional, UI, physical, logical cases, etc. Test cases compare available resources and current conditions with desired outcomes and determine if the functionality can be released or not.

Traceability Matrix. This software testing documentation maps test cases and their requirements. All entries have their custom IDs — team members and stakeholders can track the progress of any tasks by simply entering its ID to the search.

External documentation collects information from inner documentation but also emphasize on providing a visual data representation — graphs, diagrams, etc.

External reports — these documents collect information on test results and can describe an entire project or a particular piece of functionality.

Test summary report — the file with final test results and findings, presented to stakeholders.

Bug reports — such files keep track of newly encountered bugs and their fixes. We prefer to keep our bug documentation numbered, so it's easier to mention them in further documentation. Reports are concise and focus on offering tangible solutions. Sometimes, bug reports can only include issue description, if the team hasn't yet found the best approach to fixing the problem.

The combination of internal and external documentation is the key to a deep understanding of all testing processes. Although stakeholders typically have access to the majority of documentation, they mostly work with external files, since they are more concise and tackle tangible issues and results. Internal files, on the other hand, are used by team members to optimize the testing process.

Black Box testers don't care about Unit Testing. Their main goal is to validate the application against the requirements without going into the implementation details.

But as a curiosity or Out of the box thinking, have you ever wondered how developers test their own code? What method do they use to test before releasing code for testing? How is dev-testing important in an agile process? The answer to all this is Unit Testing. I want to educate you on the importance of Unit Testing so that development and testing teams can work more collaboratively to design, test and release an excellent application.

Unit Testing is not a new concept. It's been there since the early days of programming. Usually, developers and sometimes White box testers write Unit tests to improve code quality by verifying each and every unit of the code used to implement functional requirements (aka test driven development TDD or test-first development).

Most of us might know the classic definition of Unit Testing

“Unit Testing is the method of verifying the smallest piece of testable code against its purpose.” If the purpose or requirement failed then the unit test has failed.

In simple words, Unit Testing means – writing a piece of code (unit test) to verify the code (unit) written for implementing requirements.



Fig 7.1

Unit Testing is used to design robust software components that help maintain code and eliminate the issues in code units. We all know the importance of finding and fixing defects in the early stage of the software development cycle. Unit Testing serves the same purpose.

Unit Testing is an integral part of the agile software development process. When a nightly build run unit test suite should run and report should be generated. If any of the unit tests have failed then the QA team should not accept that build for verification.

If we set this as a standard process, many defects would be caught in the early development cycle, saving much testing time.

I know many developers hate to write unit tests. They either ignore or write bad unit test cases due to tight scheduled or lack of seriousness (yea they write empty unit tests, so 100% of them pass successfully ☐). It's important to write good unit tests or don't write them at all. It's even more important to provide sufficient time and a supportive environment for real benefits.

Testing can be done in the early phases of the software development lifecycle when other modules may not be available for integration.

Fixing an issue in Unit Testing can fix many other issues occurring in later

development and testing stages.

Cost of fixing a defect found in Unit Testing is very less than the one found in the system or acceptance testing.

Code coverage can be measured

Fewer bugs in the System and Acceptance testing

Code completeness can be demonstrated using unit tests. This is more useful in the agile process. Testers don't get the functional builds to test until integration is completed. Code completion cannot be justified by showing that you have written and checked in the code. But running Unit tests can demonstrate code completeness.

Expect robust design and development as developers write test cases by understanding the specifications first.

Easily identify who broke the build.

Saves development time: Code completion may take more time but due to decreased defect count overall development time can be saved.

7.1 GENERAL

Unit Testing frameworks are mostly used to help write unit tests quickly and easily. Most of the programming languages do not support unit testing with the inbuilt compiler. Third-party open source and commercial tools can be used to make unit testing even more fun.

List of popular Unit Testing tools for different programming languages:

Java framework – Junit

PHP framework – PHPUnit

C++ frameworks – UnitTest++ and Google C++

.NET framework – Nunit

Python framework – py.test

may slow you down at first.

Functional Testing types include:

1. Unit Testing
2. Integration Testing
3. System Testing
4. Sanity Testing
5. Smoke Testing
6. Interface Testing
7. Regression Testing
8. Beta/Acceptance Testing
9. Non-functional Testing types

include: Performance Testing

- Load Testing
- Stress Testing
- Volume Testing
- Security Testing
- Compatibility Testing
- Install Testing
- Recovery Testing
- Reliability Testing
- Usability Testing
- Compliance Testing
- Localization Testing

7.3 UNIT TESTING

UNIT TESTING is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented

programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or

derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.)

Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.

A unit can be almost anything you want it to be -- a line of code, a method, or a class. Generally though, smaller is better. Smaller tests give you a much more granular view of how your code is performing. There is also the practical aspect that when you test very small units, your tests can be run fast; like a thousand tests in a second fast.

Black Box testers don't care about Unit Testing. Their main goal is to validate the application against the requirements without going into the implementation details.

Unit Testing is not a new concept. It's been there since the early days of programming. Usually, developers and sometimes White box testers write Unit tests to improve code quality by verifying each and every unit of the code used to implement functional requirements (aka test drove development TDD or test-first development).

Most of us might know the classic definition of Unit Testing –

“Unit Testing is the method of verifying the smallest piece of testable code against its purpose.” If the purpose or requirement failed then the unit test has failed.

In simple words, Unit Testing means – writing a piece of code (unit test) to verify the code (unit) written for implementing requirements.

7.4 BLACKBOX TESTING

During functional testing, testers verify the app features against the user specifications. This is completely different from testing done by developers which is unit testing. It checks whether the code works as expected. Because unit testing

focuses on the internal structure of the code, it is called the white box testing. On the other hand, functional testing checks app's functionalities without looking at the internal structure of the code, hence it is called black box testing. Despite how flawless the various individual code components may be, it is essential to check that the app is functioning

as expected, when all components are combined. Here you can find a detailed comparison between functional testing vs unit testing.

7.5 INTEGRATION TESTING

INTEGRATION TESTING is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

integration testing: Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. See also component integration testing, system integration testing.

component integration testing: Testing performed to expose defects in the interfaces and interaction between integrated components. System integration testing: Testing the integration of systems and packages; testing interfaces to external organizations (e.g. Electronic Data Interchange, Internet).

Integration tests determine if independently developed units of software work correctly when they are connected to each other. The term has become blurred even by the diffuse standards of the software industry, so I've been wary of using it in my writing. In particular, many people assume integration tests are necessarily broad in scope, while they can be more effectively done with a narrower scope.

As often with these things, it's best to start with a bit of history. When I first learned about integration testing, it was in the 1980's and the waterfall was the dominant influence of software development thinking. In a larger project, we would have a design phase that would specify the interface and behavior of the various modules in the system. Modules would then be assigned to developers to program. It was

not unusual for one programmer to be responsible for a single module, but this would be big enough that it could take months to build it. All this work was done in isolation, and when the programmer believed it was finished they would hand it over to QA for testing.

Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems. Integration testing is a key aspect of software testing.

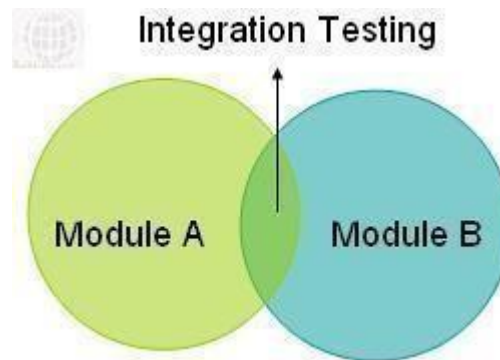


Fig 7.3

7.6 SYSTEM TESTING

SYSTEM TESTING is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements. System Testing means testing the system as a whole. All the modules/components are integrated in order to verify if the system works as expected or not. System Testing is done after Integration Testing. This plays an important role in delivering a high-quality product.

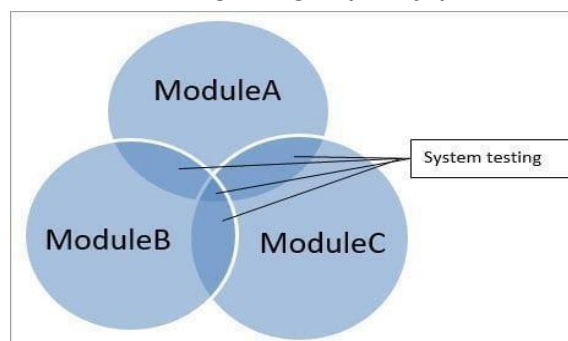


Fig 7.4

System testing is a method of monitoring and assessing the behaviour of the complete and fully-integrated software product or system, on the basis of pre-decided specifications and functional requirements. It is a solution to the question "whether the complete system functions in accordance to its pre-defined requirements?"

It's comes under black box testing i.e. only external working features of the software are evaluated during this testing. It does not requires any internal knowledge of the coding, programming, design, etc., and is completely based on users-perspective.

A black box testing type, system testing is the first testing technique that carries out the task of testing a software product as a whole. This System testing tests the integrated system and validates whether it meets the specified requirements of the client.

System testing is a process of testing the entire system that is fully functional, in order to ensure the system is bound to all the requirements provided by the client in the form of the functional specification or system specification documentation. In most cases, it is done next to the Integration testing, as this testing should be covering the end-to-end system's actual routine. This type of testing requires a dedicated Test Plan and other test documentation derived from the system specification document that should cover both software and hardware requirements. By this test, we uncover the errors. It ensures that all the system works as expected. We check System performance and functionality to get a quality product. System testing is nothing but testing the system as a whole. This testing checks complete end-to-end scenario as per the customer's point of view. Functional and Non-Functional tests also done by System testing. All things are done to maintain trust within the development that the system is defect-free and bug-free. System testing is also intended to test hardware/software requirements specifications. System testing is more of a limited type of testing; it seeks to detect both defects within the "inter-assemblages".

7.7 SANITY TESTING

Sanity Testing is done when as a QA we do not have sufficient time to run all the test cases, be it Functional Testing, UI, OS or Browser Testing. Sanity testing is a subset of regression testing. After receiving the software build, sanity testing is performed to ensure that the code changes introduced are working as expected. This testing is a checkpoint to determine if testing for the build can proceed or not.

The main purpose of this testing is to determine that the changes or the proposed functionality are working as expected. If the sanity test fails, the build is rejected by the testing team to save time and money. It is performed only after the build has cleared the smoke test and been accepted by the Quality Assurance team for further testing. The focus of the team during this testing process is to validate the functionality of the application and not detailed testing. Let's assume that in an e-commerce project, there are mainly five modules- like login page, home page, user profile page, new user creation, etc. There is a defect in the login page when the password field accepts less than four alpha numeric characters and the requirement mentions that this password field should not be below eight characters. So, the bug is reported by the testing team to the development team to resolve it. When the development team fixes the defect and passes it to the testing team. Then the testing team checks if the changes done are working fine. It also checks if it does not have an impact on other related functionalities. Now there is a functionality to update the password in the user profile page. As part of the sanity check, we would need to validate the login page as well as the profile page to ensure that the checks are working fine at both the places. Sanity tests are generally performed on build where the production deployment is required immediately - like a critical bug fix.

Smoke Testing is done to make sure if the build we received from the development team is testable or not. It is also called as "Day 0" check. It is done at the "build level".

It helps not to waste the testing time to simply testing the whole application when the key features don't work or the key bugs have not been fixed yet. Here our focus will be on primary and core application work flow.

To conduct smoke testing, we don't write test cases. We just pick the necessary test cases from already written test cases.

Do we really write test cases for all testing types? Here in this article, we have given clear idea on choosing testing types to write test cases.

As mentioned earlier, here in Smoke Testing, our main focus will be on core application work flow. So we pick the test cases from our test suite which cover major functionality of the application. In general, we pick minimal number of test cases that won't take more than half an hour to execute.

After receiving a Software build with the minor issues fixes in code or functionality, Sanity testing is carry out to check whether the bugs reported in previous build are fixed & there is regression introduced due to these fixes i.e. not breaking any previously working functionality. The main aim of Sanity testing to check the planned functionality is working as expected. Instead of doing whole regression testing the Sanity testing is perform.

Sanity tests helps to avoid wasting time and cost involved in testing if the build is failed. Tester should reject the build upon build failure.

After completion of regression testing the Sanity testing is started to check the defect fixes & changes done in the software application is not breaking the core functionality of the software. Typically this is done nearing end of SDLC i.e. while releasing the software. You can say that sanity testing is a subset of acceptance testing. We can also say Tester Acceptance Testing for Sanity testing.

7.8 REGRESSION TESTING

Regression Testing is a type of testing that is done to verify that a code change in the software does not impact the existing functionality of the product. This is to make sure the product works fine with new functionality, bug fixes or any change in the existing feature. Previously executed test cases are re-executed in order to verify the impact of change.

Regression Testing is a Software Testing type in which test cases are re-executed in order to check whether the previous functionality of the application is working fine and the new changes have not introduced any new bugs.

This test can be performed on a new build when there is a significant change in the original functionality that too even in a single bug fix.

For regression testing to be effective, it needs to be seen as one part of a comprehensive testing methodology that is cost-effective and efficient while still incorporating enough variety—such as well-designed frontend UI automated tests alongside targeted unit testing, based on smart risk prioritization—to prevent any aspects of your software applications from going unchecked. These days, many Agile work environments employing workflow practices such as XP (Extreme Programming), RUP (Rational Unified Process), or Scrum appreciate regression testing as an essential aspect of a dynamic, iterative development and deployment schedule.

But no matter what software development and quality-assurance process your organization uses, if you take the time to put in enough careful planning up front, crafting a clear and diverse testing strategy with automated regression testing at its core, you can help prevent projects from going over budget, keep your team on track, and, most importantly, prevent unexpected bugs from damaging your products and your company's bottom line.

Performance testing is the practice of evaluating how a system performs in terms of responsiveness and stability under a particular workload. Performance tests are typically executed to examine speed, robustness, reliability, and application size.

The process incorporates “performance” indicators such as:

- Browser, page, and network response times
- Server request processing times
- Acceptable concurrent user volumes
- Processor memory consumption; number and type of errors that might be encountered with app

Performance testing gathers all the tests that verify an application's speed, robustness, reliability, and correct sizing. It examines several indicators such as a browser, page and network response times, server query processing time, number of acceptable concurrent users architected, CPU memory consumption, and number/type of errors which may be encountered when using an application.

Performance testing is the testing that is performed to ascertain how the components of a system are performing under a certain given situation.

Resource usage, scalability, and reliability of the product are also validated under this testing. This testing is the subset of performance engineering, which is focused on addressing performance issues in the design and architecture of a software product.



Fig 7.5

Software Performance testing is type of testing perform to determine the performance of system to major the measure, validate or verify quality attributes of the system like responsiveness, Speed, Scalability, Stability under variety of load conditions. The system is tested under a mixture of load conditions and check the time required responding by the system under varying workloads. Software performance testing involves the testing of application under test to ensure that application is working as expected under variety of load conditions. The goal of performance testing is not only find the bugs in the system but also eliminate the performance bottlenecks from the system.

Load Testing is type of performance testing to check system with constantly increasing the load on the system until the time load is reaches to its threshold value. Here Increasing load means increasing number of concurrent users, transactions & check the behavior of application under test. It is normally carried out underneath controlled environment in order to distinguish between two different systems. It is also called as “Endurance testing” and “Volume testing”. The main purpose of load testing is to monitor the response time and staying power of application when system is performing well under heavy load. Load testing comes under the Non Functional Testing & it is designed to test the non-functional requirements of a software application.

Load testing is perform to make sure that what amount of load can be withstand the application under test. The successfully executed load testing is only if the specified test cases are executed without any error in allocated time.

Simple examples of load testing:

- Testing printer by sending large job.
- Editing a very large document for testing of word processor
- Continuously reading and writing data into hard disk.
- Running multiple applications simultaneously on server.
- Testing of mail server by accessing thousands of mailboxes.
- In case of zero-volume testing & system fed with zero load.

CHAPTE R 8

CONCLUSION & FUTURE ENCHANCEMENT

8.1 CONCLUSION

We presented a novel ridesharing service, CommuteShare, which is able to match rideshare companions with a high accuracy for daily commuters. Specifically, CommuteShare significantly outperforms the baseline methods in improving both riders' satisfaction and social benefits: 1) CommuteShare achieves higher rideshare accuracy (an average daily rideshare accuracy than the baseline methods; 2) CommuteShare can reduce of commuting private vehicles during morning peak hours of workdays.

8.2 FUTURE ENHANCEMENT

Future works may include: 1) extending the algorithm to enable dynamic ridesharing in evening peak hours where there are many complicated alternatives for the commuters rather than going home directly (e.g., going shopping, dining out, visiting a bar, etc.); and 2) employing deep neural networks to uncover the complexity in ridesharing

APPENDIX – I SAMPLE CODING

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
import seaborn as sns
from matplotlib import cm
from matplotlib import colors
from bokeh.plotting import show as show
from bokeh.layouts import row as row
from bokeh.models import ColumnDataSource, DataRange1d, HoverTool,
PanTool, WheelZoomTool
from bokeh.models.widgets import DataTable, TableColumn
from bokeh.models.widgets import DateFormatter, StringFormatter
from sklearn.cluster import MiniBatchKMeans as ACluster
from pandas import pandas as pd
from sklearn import decomposition as decomposition
from scipy import stats as stats

matplotlib.style.use('fivethirtyeight')
matplotlib.rcParams['font.size'] = 12
matplotlib.rcParams['figure.figsize'] = (10,10)
plt.style.use('ggplot')
plt.rc('font', size=12)
COLOR_DARK = "#212121"
COLOR_YELLOW = "#ffb700"

plt.register_cmap(cmap=colors.LinearSegmentedColormap.from_list(
    'TaxiYellow', ['#ffffff', COLOR_YELLOW]))
dataDir = './'
```

```

taxiDB = pd.read_csv(dataDir + 'train.csv')
taxiDB.shape
taxiDB.info
taxiDB.head()
taxiDB.tail()
taxiDB.sample(25)
allLat = np.array(list(taxiDB['pickup_latitude']) + list(taxiDB['dropoff_latitude']))
allLong = np.array(list(taxiDB['pickup_longitude']) +
list(taxiDB['dropoff_longitude']))
longLimits = [np.percentile(allLong, 0.3), np.percentile(allLong, 99.7)]
latLimits = [np.percentile(allLat , 0.3), np.percentile(allLat , 99.7)]
durLimits = [np.percentile(taxiDB['trip_duration'], 0.4),
np.percentile(taxiDB['trip_duration'], 99.7)]
taxiDB = taxiDB[(taxiDB['pickup_latitude'] >= latLimits[0] ) &
(taxiDB['pickup_latitude'] <= latLimits[1]) ]
taxiDB = taxiDB[(taxiDB['dropoff_latitude'] >= latLimits[0] ) &
(taxiDB['dropoff_latitude'] <= latLimits[1]) ]
taxiDB = taxiDB[(taxiDB['pickup_longitude'] >= longLimits[0]) &
(taxiDB['pickup_longitude'] <= longLimits[1])]
taxiDB = taxiDB[(taxiDB['dropoff_longitude'] >= longLimits[0]) &
(taxiDB['dropoff_longitude'] <= longLimits[1])]
taxiDB = taxiDB[(taxiDB['trip_duration']    >= durLimits[0] ) &
(taxiDB['trip_duration']    <= durLimits[1]) ]
taxiDB = taxiDB.reset_index(drop=True)
allLat = np.array(list(taxiDB['pickup_latitude']) + list(taxiDB['dropoff_latitude']))
allLong = np.array(list(taxiDB['pickup_longitude']) +
list(taxiDB['dropoff_longitude']))
medianLat = np.percentile(allLat,50)
medianLong = np.percentile(allLong,50)
latMultiplier = 111.32
longMultiplier = np.cos(medianLat*(np.pi/180.0)) * 111.32

```

```

taxiDB['duration [min]'] = taxiDB['trip_duration']/60.0
taxiDB['src lat [km]'] = latMultiplier * (taxiDB['pickup_latitude'] - medianLat)
taxiDB['src long [km]'] = longMultiplier * (taxiDB['pickup_longitude'] -
medianLong)
taxiDB['dst lat [km]'] = latMultiplier * (taxiDB['dropoff_latitude'] - medianLat)
taxiDB['dst long [km]'] = longMultiplier * (taxiDB['dropoff_longitude'] -
medianLong)
allLat = np.array(list(taxiDB['src lat [km]']) + list(taxiDB['dst lat [km]']))
allLong = np.array(list(taxiDB['src long [km]']) + list(taxiDB['dst long [km]']))
fig, axArray = plt.subplots(nrows=1,ncols=3,figsize=(13,4))
axArray[0].hist(taxiDB['duration [min]'],80);
axArray[0].set_xlabel('trip duration [min]'); axArray[0].set_ylabel('counts')
axArray[1].hist(allLat ,80); axArray[1].set_xlabel('latitude [km]')
axArray[2].hist(allLong,80); axArray[2].set_xlabel('longitude [km]')
taxiDB['log duration'] = np.log1p(taxiDB['duration [min]'])
taxiDB['euclidian distance'] = np.sqrt((taxiDB['src lat [km]'] - taxiDB['dst lat [km]']
)**2 +
                                     (taxiDB['src long [km]'] - taxiDB['dst long [km]'])**2)
fig, axArray = plt.subplots(nrows=1,ncols=2,figsize=(13,6))
axArray[0].scatter(taxiDB['euclidian distance'], taxiDB['duration
[min]'],c='r',s=5,alpha=0.01);
axArray[0].set_xlabel('Aerial Euclidian Distance [km]');
axArray[0].set_ylabel('Duration [min]')
axArray[0].set_xlim(taxiDB['euclidian distance'].min(),taxiDB['euclidian
distance'].max())
axArray[0].set_ylim(taxiDB['duration [min]'].min(),taxiDB['duration [min]'].max())
axArray[0].set_title('trip Duration vs Aerial trip Distance')

axArray[1].scatter(taxiDB['euclidian distance'], taxiDB['log
duration'],c='r',s=5,alpha=0.01);
axArray[1].set_xlabel('Aerial Euclidian Distance [km]');

```

```

axArray[1].set_ylabel('log(1+Duration) [log(min)]')
axArray[1].set_xlim(taxiDB['euclidian distance'].min(),taxiDB['euclidian
distance'].max())
axArray[1].set_ylim(taxiDB['log duration'].min(),taxiDB['log duration'].max())
axArray[1].set_title('log of trip Duration vs Aerial trip Distance')
imageSize = (700,700)
longRange = [-5,19]
latRange = [-13,11]
allLatInds = imageSize[0] - (imageSize[0] * (allLat - latRange[0]) / (latRange[1]
- latRange[0])) .astype(int)
allLongInds = (imageSize[1] * (allLong - longRange[0]) / (longRange[1]
- longRange[0])).astype(int)

fig, ax = plt.subplots(nrows=1,ncols=1,figsize=(12,12))
ax.imshow(np.log(locationDensityImage+1),cmap='hot')
ax.set_axis_off()
imageSizeMan = (720,480)
latRangeMan = [-8,10]
longRangeMan = [-5,7]
indToKeep = np.logical_and(allLat > latRangeMan[0], allLat < latRangeMan[1])
indToKeep = np.logical_and(indToKeep, np.logical_and(allLong >
longRangeMan[0], allLong < longRangeMan[1]))
allLatMan = allLat[indToKeep]
allLongMan = allLong[indToKeep]
allLatIndsMan = (imageSizeMan[0]-1) - (imageSizeMan[0] * (allLatMan -
latRangeMan[0])
/ (latRangeMan[1] -
latRangeMan[0])).astype(int)
allLongIndsMan = (imageSizeMan[1] * (allLongMan -
longRangeMan[0])
/ (longRangeMan[1] -

```

```

longRangeMan[0])).astype(int)
locationDensityImageMan = np.zeros(imageSizeMan)
for latInd, longInd in zip(allLatIndsMan,allLongIndsMan):
    locationDensityImageMan[latInd,longInd] += 1
fig, ax = plt.subplots(nrows=1,ncols=1,figsize=(12,18))
ax.imshow(np.log(locationDensityImageMan+1),cmap='hot')
ax.set_axis_off()
pickupTime = pd.to_datetime(taxiDB['pickup_datetime'])
taxiDB['src hourOfDay'] = (pickupTime.dt.hour*60.0 + pickupTime.dt.minute) /
60.0
taxiDB['dst hourOfDay'] = taxiDB['src hourOfDay'] + taxiDB['duration [min]'] / 60.0
taxiDB['dayOfWeek'] = pickupTime.dt.weekday
taxiDB['hourOfWeek'] = taxiDB['dayOfWeek']*24.0 + taxiDB['src hourOfDay']
taxiDB['monthOfYear'] = pickupTime.dt.month
taxiDB['dayOfYear'] = pickupTime.dt.dayofyear
taxiDB['weekOfYear'] = pickupTime.dt.weekofyear
taxiDB['hourOfYear'] = taxiDB['dayOfYear']*24.0 + taxiDB['src hourOfDay']
tripAttributes = np.array(taxiDB.loc[:,['src lat [km]','src long [km]','dst lat [km]','dst
long [km]','duration [min]']])
meanTripAttr = tripAttributes.mean(axis=0)
stdTripAttr = tripAttributes.std(axis=0)
tripAttributes = stats.zscore(tripAttributes, axis=0)
numClusters = 80
aClusterModel = ACluster(n_clusters=numClusters, batch_size=120000,
n_init=100, random_state=1)
clusterInds = aClusterModel.fit_predict(tripAttributes)
clusterTotalCounts, _ = np.histogram(clusterInds, bins=numClusters)
sortedClusterInds = np.flipud(np.argsort(clusterTotalCounts))
plt.figure(figsize=(12,4)); plt.title('Cluster Histogram of all trip')
plt.bar(range(1,numClusters+1),clusterTotalCounts[sortedClusterInds])
plt.ylabel('Frequency [counts]'); plt.xlabel('Cluster index (sorted by cluster

```

```

frequency))
plt.xlim(0,numClusters+1)
def ConvertToImageCoords(latCoord, longCoord, latRange, longRange,
imageSize):
    latInds = imageSize[0] - (imageSize[0] * (latCoord - latRange[0]) /
(latRange[1] - latRange[0]) ).astype(int)
    longInds = (imageSize[1] * (longCoord - longRange[0]) /
(longRange[1] - longRange[0])).astype(int)

    return latInds, longInds
templateTrips = aClusterModel.cluster_centers_ *
np.tile(stdTripAttr,(numClusters,1)) + np.tile(meanTripAttr,(numClusters,1))
srcCoords = templateTrips[:,2:]
dstCoords = templateTrips[:,2:4]
srcImCoords = ConvertToImageCoords(srcCoords[:,0],srcCoords[:,1], latRange,
longRange, imageSize)
dstImCoords = ConvertToImageCoords(dstCoords[:,0],dstCoords[:,1], latRange,
longRange, imageSize)
plt.figure(figsize=(12,12))
plt.imshow(np.log(locationDensityImage+1),cmap='hot'); plt.grid('off')
plt.scatter(srcImCoords[1],srcImCoords[0],c='m',s=200,alpha=0.8)
plt.scatter(dstImCoords[1],dstImCoords[0],c='g',s=200,alpha=0.8)

for i in range(len(srcImCoords[0])):
    plt.arrow(srcImCoords[1][i],srcImCoords[0][i],
dstImCoords[1][i]-srcImCoords[1][i], dstImCoords[0][i]-srcImCoords[0][i],
            edgecolor='c', facecolor='c',
width=0.8,alpha=0.4,head_width=10.0,head_length=10.0,length_includes_head
=True)
hoursOfDay = np.sort(taxiDB['src hourOfDay'].astype(int).unique())
clusterDistributionHourOfDay_weekday =

```

```

np.zeros((len(hoursOfDay),numClusters))
for k, hour in enumerate(hoursOfDay):
    slectedInds = (taxiDB['src hourOfDay'].astype(int) == hour) &
(taxiDB['dayOfWeek'] <= 4)
    currDistribution, _ = np.histogram(clusterInds[slectedInds], bins=numClusters)
    clusterDistributionHourOfDay_weekday[k,:] =
currDistribution[sortedClusterInds]

```

```

fig, ax = plt.subplots(nrows=1,ncols=1,figsize=(12,6))
ax.set_title('Trip Distribution during Weekdays', fontsize=12)
ax.imshow(clusterDistributionHourOfDay_weekday); ax.grid('off')
ax.set_xlabel('Trip Cluster'); ax.set_ylabel('Hour of Day')
ax.annotate('Silent Nights', color='r', fontsize=15, xy=(52, 2), xytext=(58, 1.75),
            arrowprops=dict(facecolor='red', shrink=0.03))

```

```

hoursOfDay = np.sort(taxiDB['src hourOfDay'].astype(int).unique())
clusterDistributionHourOfDay_weekend =
np.zeros((len(hoursOfDay),numClusters))
for k, hour in enumerate(hoursOfDay):
    slectedInds = (taxiDB['src hourOfDay'].astype(int) == hour) &
(taxiDB['dayOfWeek'] >= 5)
    currDistribution, _ = np.histogram(clusterInds[slectedInds], bins=numClusters)
    clusterDistributionHourOfDay_weekend[k,:] =
currDistribution[sortedClusterInds]

```

```

fig, ax = plt.subplots(nrows=1,ncols=1,figsize=(12,6))
ax.set_title('Trip Distribution during Weekends', fontsize=12)
ax.imshow(clusterDistributionHourOfDay_weekend); ax.grid('off')
ax.set_xlabel('Trip Cluster'); ax.set_ylabel('Hour of Day')
ax.annotate('Party Nights', color='r', fontsize=15, xy=(52, 2), xytext=(58, 1.75),
            arrowprops=dict(facecolor='red', shrink=0.03))
ax.annotate('Late Mornings', color='r', fontsize=15, xy=(45, 10), xytext=(58,

```

```

9.75),
    arrowprops=dict(facecolor='red', shrink=0.03))
daysOfWeek = np.sort(taxiDB['dayOfWeek'].unique())
clusterDistributionDayOfWeek = np.zeros((len(daysOfWeek),numClusters))
for k, day in enumerate(daysOfWeek):
    slectedInds = taxiDB['dayOfWeek'] == day
    currDistribution, _ = np.histogram(clusterInds[slectedInds], bins=numClusters)
    clusterDistributionDayOfWeek[k,:] = currDistribution[sortedClusterInds]
plt.figure(figsize=(12,5)); plt.title('Trip Distribution throughout the Week')
plt.imshow(clusterDistributionDayOfWeek); plt.grid('off')
plt.xlabel('Trip Cluster'); plt.ylabel('Day of Week')
daysOfYear = taxiDB['dayOfYear'].unique()
daysOfYear = np.sort(daysOfYear)
clusterDistributionDayOfYear = np.zeros((len(daysOfYear),numClusters))
for k, day in enumerate(daysOfYear):
    slectedInds = taxiDB['dayOfYear'] == day
    currDistribution, _ = np.histogram(clusterInds[slectedInds], bins=numClusters)
    clusterDistributionDayOfYear[k,:] = currDistribution[sortedClusterInds]
fig, ax = plt.subplots(nrows=1,ncols=1,figsize=(10,16))
ax.set_title('Trip Distribution throughout the Year', fontsize=12)
ax.imshow(clusterDistributionDayOfYear); ax.grid('off')
ax.set_xlabel('Trip Cluster'); ax.set_ylabel('Day of Year')
ax.annotate('Large Snowstorm', color='r', fontsize=15 ,xy=(35, 21), xytext=(50,
17),
    arrowprops=dict(facecolor='red', shrink=0.03))
ax.annotate('Memorial Day', color='r', fontsize=15, xy=(35, 151), xytext=(50,
157),
    arrowprops=dict(facecolor='red', shrink=0.03))
hoursOfYear = np.sort(taxiDB['hourOfYear'].astype(int).unique())
clusterDistributionHourOfYear =
np.zeros((len(range(hoursOfYear[0],hoursOfYear[-1])),numClusters))

```

```

dayOfYearVec = np.zeros(clusterDistributionHourOfYear.shape[0])
weekdayVec    = np.zeros(clusterDistributionHourOfYear.shape[0])
weekOfYearVec = np.zeros(clusterDistributionHourOfYear.shape[0])
for k, hour in enumerate(hoursOfYear):
    slectedInds = taxiDB['hourOfYear'].astype(int) == hour
    currDistribution, _ = np.histogram(clusterInds[slectedInds], bins=numClusters)
    clusterDistributionHourOfYear[k,:] = currDistribution[sortedClusterInds]

    dayOfYearVec[k] = taxiDB[slectedInds]['dayOfYear'].mean()
    weekdayVec[k]    = taxiDB[slectedInds]['dayOfWeek'].mean()
    weekOfYearVec[k] = taxiDB[slectedInds]['weekOfYear'].mean()
numComponents = 3
TripDistributionPCAModel =
decomposition.PCA(n_components=numComponents,whiten=True,
random_state=1)
compactClusterDistributionHourOfYear =
TripDistributionPCAModel.fit_transform(clusterDistributionHourOfYear)
listOfFullWeeks = []
for uniqueVal in np.unique(weekOfYearVec):
    if (weekOfYearVec == uniqueVal).sum() == 24*7:
        listOfFullWeeks.append(uniqueVal)
weeklyTraces = np.zeros((24*7,numComponents,len(listOfFullWeeks)))
for k, weekInd in enumerate(listOfFullWeeks):
    weeklyTraces[:,k] = compactClusterDistributionHourOfYear[weekOfYearVec
== weekInd,:]
fig, axArray = plt.subplots(nrows=numComponents,ncols=1,sharex=True,
figsize=(12,12))
fig.suptitle('PCA coefficients during the Week', fontsize=25)
for PC_coeff in range(numComponents):
    meanTrace = weeklyTraces[:,PC_coeff,:].mean(axis=1)
    axArray[PC_coeff].plot(weeklyTraces[:,PC_coeff,:],'y',linewidth=1.5)

```

```

axArray[PC_coeff].plot(meanTrace,'k',linewidth=2.5)
axArray[PC_coeff].set_ylabel('PC %d coeff' %(PC_coeff+1))
axArray[PC_coeff].vlines([0,23,47,71,95,119,143,167],
weeklyTraces[:,PC_coeff,:].min(), weeklyTraces[:,PC_coeff,:].max(), colors='r',
lw=2)

axArray[PC_coeff].set_xlabel('hours since start of week')
axArray[PC_coeff].set_xlim(-0.9,24*7-0.1)
listOfFullWeekdays = []
listOfFullWeekends = []
for uniqueVal in np.unique(dayOfYearVec):
    if (dayOfYearVec == uniqueVal).sum() == 24:
        if weekdayVec[dayOfYearVec == uniqueVal][0] <= 4:
            listOfFullWeekdays.append(uniqueVal)
        else:
            listOfFullWeekends.append(uniqueVal)
weekdayTraces = np.zeros((24,numComponents,len(listOfFullWeekdays)))
for k, dayInd in enumerate(listOfFullWeekdays):
    weekdayTraces[:,k] = compactClusterDistributionHourOfYear[dayOfYearVec
== dayInd,:]
weekendTraces = np.zeros((24,numComponents,len(listOfFullWeekends)))
for k, dayInd in enumerate(listOfFullWeekends):
    weekendTraces[:,k] = compactClusterDistributionHourOfYear[dayOfYearVec
== dayInd,:]
fig, axArray =
plt.subplots(nrows=numComponents,ncols=2,sharex=True,sharey=True,
figsize=(12,14))
fig.suptitle('PCA coefficients for weekdays and weekends', fontsize=25)
for PC_coeff in range(numComponents):
    axArray[PC_coeff][0].plot(weekdayTraces[:,PC_coeff,:],'c',linewidth=1.5)

```

```

axArray[PC_coeff][0].plot(weekdayTraces[:,PC_coeff,:].mean(axis=1),'k',linewidth
h=2.5)
    axArray[PC_coeff][0].set_ylabel('PC %d coeff' %(PC_coeff+1))

    axArray[PC_coeff][1].plot(weekendTraces[:,PC_coeff,:].mean(axis=1),'c',linewidth=1.5)

axArray[PC_coeff][1].plot(weekendTraces[:,PC_coeff,:].mean(axis=1),'k',linewidth
h=2.5)

    if PC_coeff == 0:
        axArray[PC_coeff][0].set_title('Weekday')
        axArray[PC_coeff][1].set_title('Weekend')

axArray[PC_coeff][0].set_xlabel('hours of day')
axArray[PC_coeff][1].set_xlabel('hours of day')
axArray[PC_coeff][0].set_xlim(0,23)
axArray[PC_coeff][0].set_ylim(-3.5,3.5)

axArray[2][0].annotate('Early Risers', color='r', fontsize=12, xy=(7, 2.5),
xytext=(12, 3),
                        arrowprops=dict(facecolor='red', shrink=0.03))
axArray[1][1].annotate('Party Nights', color='r', fontsize=12, xy=(3, 2.2),
xytext=(10, 3.2),
                        arrowprops=dict(facecolor='red', shrink=0.03))
axArray[2][1].annotate('Late Mornings', color='r', fontsize=12, xy=(9, 1.5),
xytext=(1, 3.5),
                        arrowprops=dict(facecolor='red', shrink=0.03))
fig, axArray = plt.subplots(nrows=numComponents,ncols=1,sharex=True,
figsize=(12,11))
fig.suptitle('Trip Distribution PCA Components', fontsize=25)
for PC_coeff in range(numComponents):

```

```

tripTemplateDistributionDifference =
TripDistributionPCAModel.components_[PC_coeff,:]*\

TripDistributionPCAModel.explained_variance_[PC_coeff]

axArray[PC_coeff].bar(range(1,numClusters+1),tripTemplateDistributionDifference)

axArray[PC_coeff].set_title('PCA %d component' %(PC_coeff+1))
axArray[PC_coeff].set_ylabel('delta frequency [counts]')

axArray[PC_coeff].set_xlabel('cluster index (sorted by cluster frequency)')
axArray[PC_coeff].set_xlim(0,numClusters+0.5)

axArray[1].hlines([-25,25], 0, numClusters+0.5, colors='r', lw=0.7)
axArray[2].hlines([-11,11], 0, numClusters+0.5, colors='r', lw=0.7)
numTopTripsToShow = 8
numBottomTripsToShow = 6
sortedTripClusters_PC2 =
np.argsort(TripDistributionPCAModel.components_[1,:])
topPositiveTripClusterInds = sortedTripClusters_PC2[-numTopTripsToShow:]
topNegativeTripClusterInds = sortedTripClusters_PC2[:numBottomTripsToShow]
allInds = np.hstack((topPositiveTripClusterInds,topNegativeTripClusterInds))
plt.figure(figsize=(12,12))
plt.imshow(np.log(locationDensityImage+1),cmap='hot'); plt.grid('off')
plt.scatter(srcImCoords[1][allInds],srcImCoords[0][allInds],c='m',s=500,alpha=0.9
)
plt.scatter(dstImCoords[1][allInds],dstImCoords[0][allInds],c='g',s=500,alpha=0.9)

for i in topPositiveTripClusterInds:
    plt.arrow(srcImCoords[1][i],srcImCoords[0][i],
dstImCoords[1][i]-srcImCoords[1][i], dstImCoords[0][i]-srcImCoords[0][i],

```

```

        edgecolor='r', facecolor='r',
width=2.8,alpha=0.9,head_width=10.0,head_length=10.0,length_includes_head
=True)

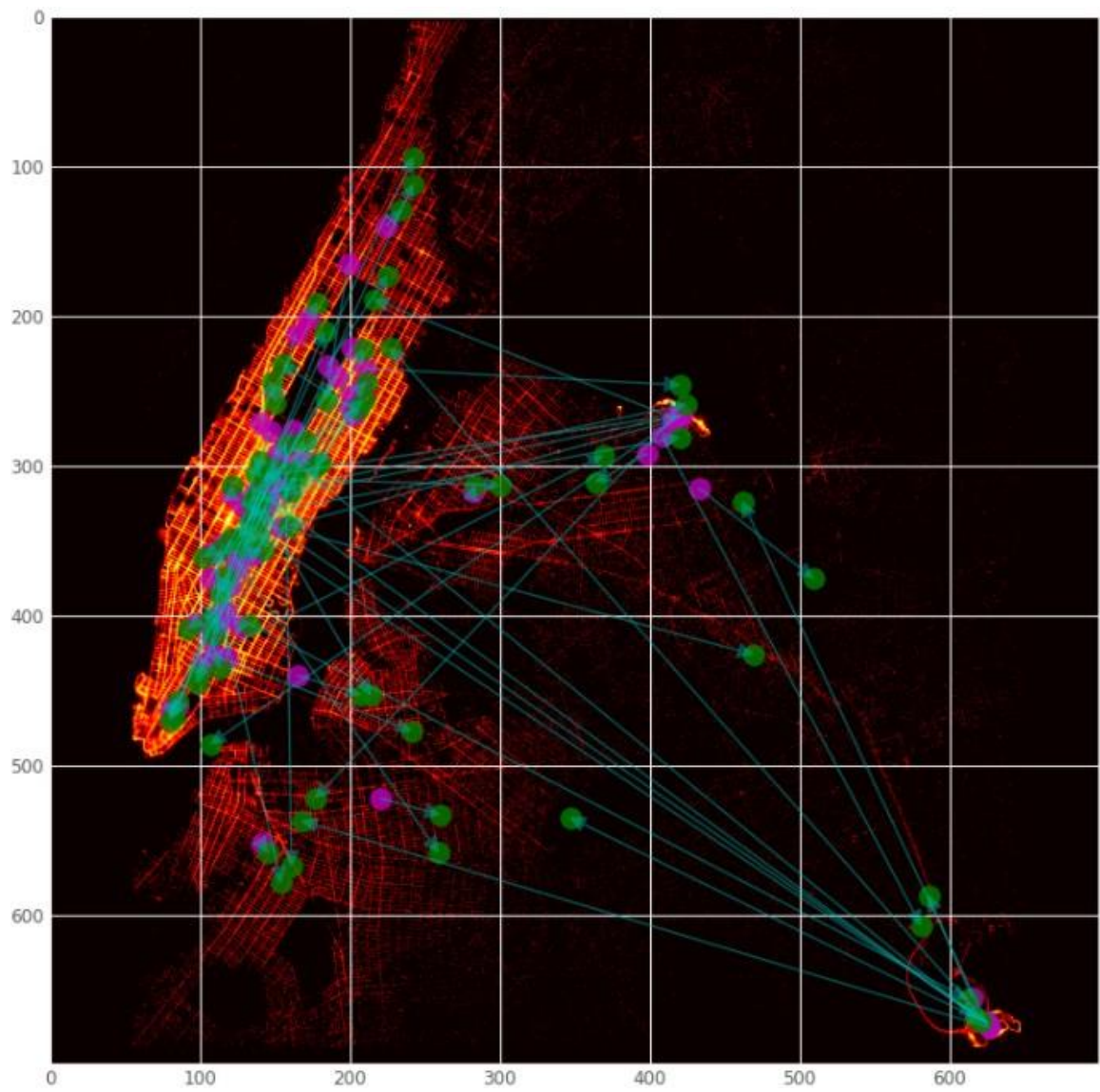
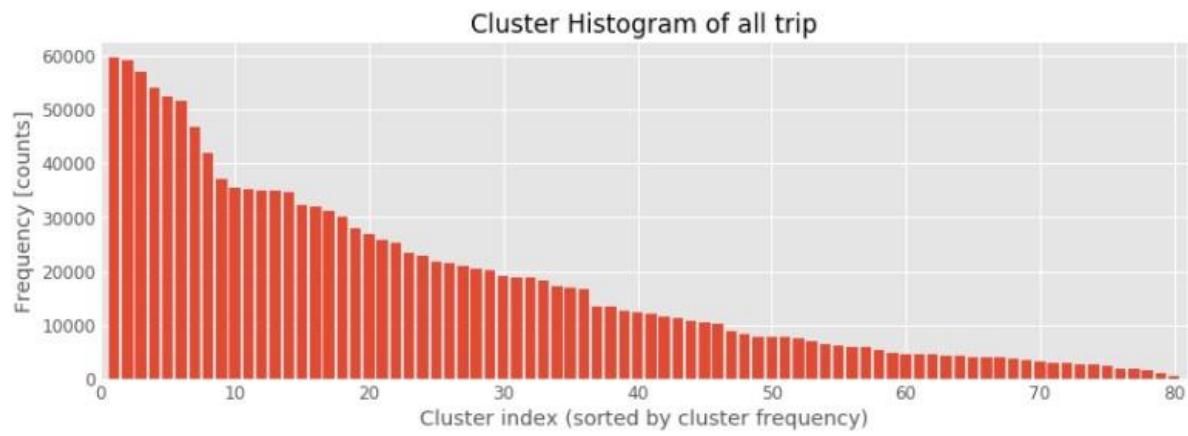
for i in topNegativeTripClusterInds:
    plt.arrow(srcImCoords[1][i],srcImCoords[0][i],
dstImCoords[1][i]-srcImCoords[1][i], dstImCoords[0][i]-srcImCoords[0][i],
        edgecolor='b', facecolor='b',
width=2.8,alpha=0.9,head_width=10.0,head_length=10.0,length_includes_head
=True)
plt.title('PC2 major Trip deviations')
numTopTripsToShow = 4
numBottomTripsToShow = 10
sortedTripClusters_PC3 =
np.argsort(TripDistributionPCAModel.components_[2,:])
topPositiveTripClusterInds = sortedTripClusters_PC3[-numTopTripsToShow:]
topNegativeTripClusterInds = sortedTripClusters_PC3[:numBottomTripsToShow]
allInds = np.hstack((topPositiveTripClusterInds,topNegativeTripClusterInds))
plt.figure(figsize=(12,12))
plt.imshow(np.log(locationDensityImage+1),cmap='hot'); plt.grid('off')
plt.scatter(srcImCoords[1][allInds],srcImCoords[0][allInds],c='m',s=500,alpha=0.9
)
plt.scatter(dstImCoords[1][allInds],dstImCoords[0][allInds],c='g',s=500,alpha=0.9)

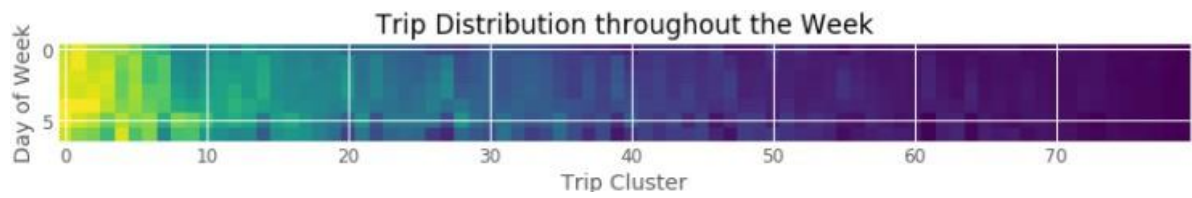
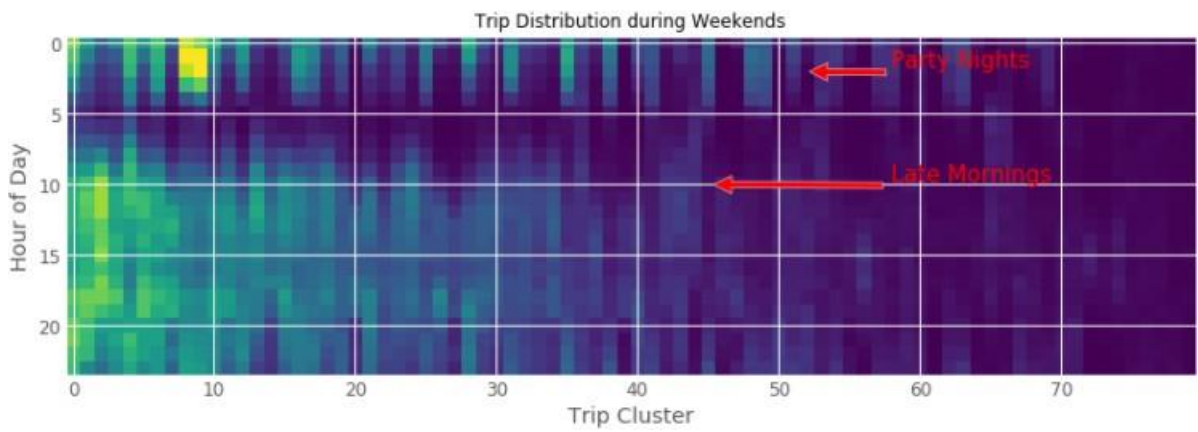
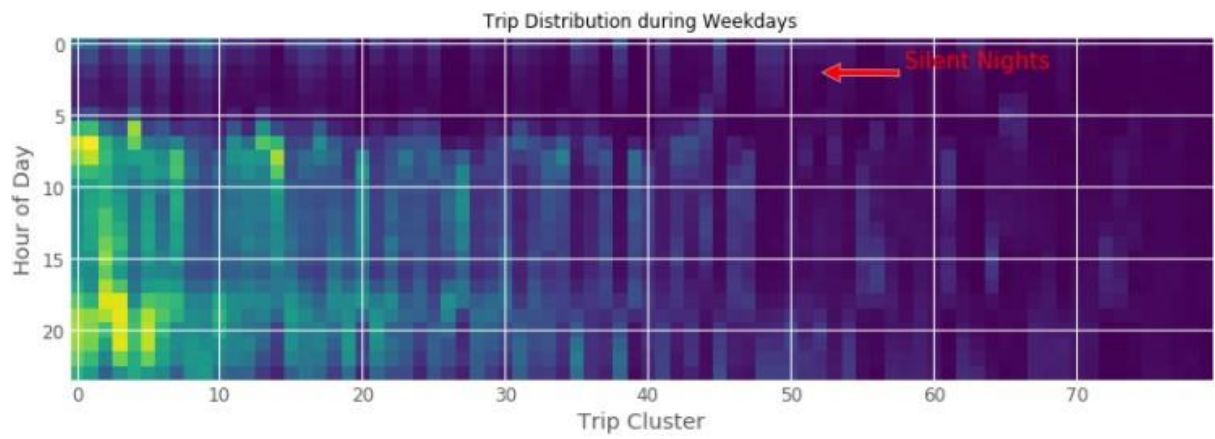
for i in topPositiveTripClusterInds:
    plt.arrow(srcImCoords[1][i],srcImCoords[0][i],
dstImCoords[1][i]-srcImCoords[1][i], dstImCoords[0][i]-srcImCoords[0][i],
        edgecolor='r', facecolor='r',
width=2.8,alpha=0.9,head_width=10.0,head_length=10.0,length_includes_head
=True)

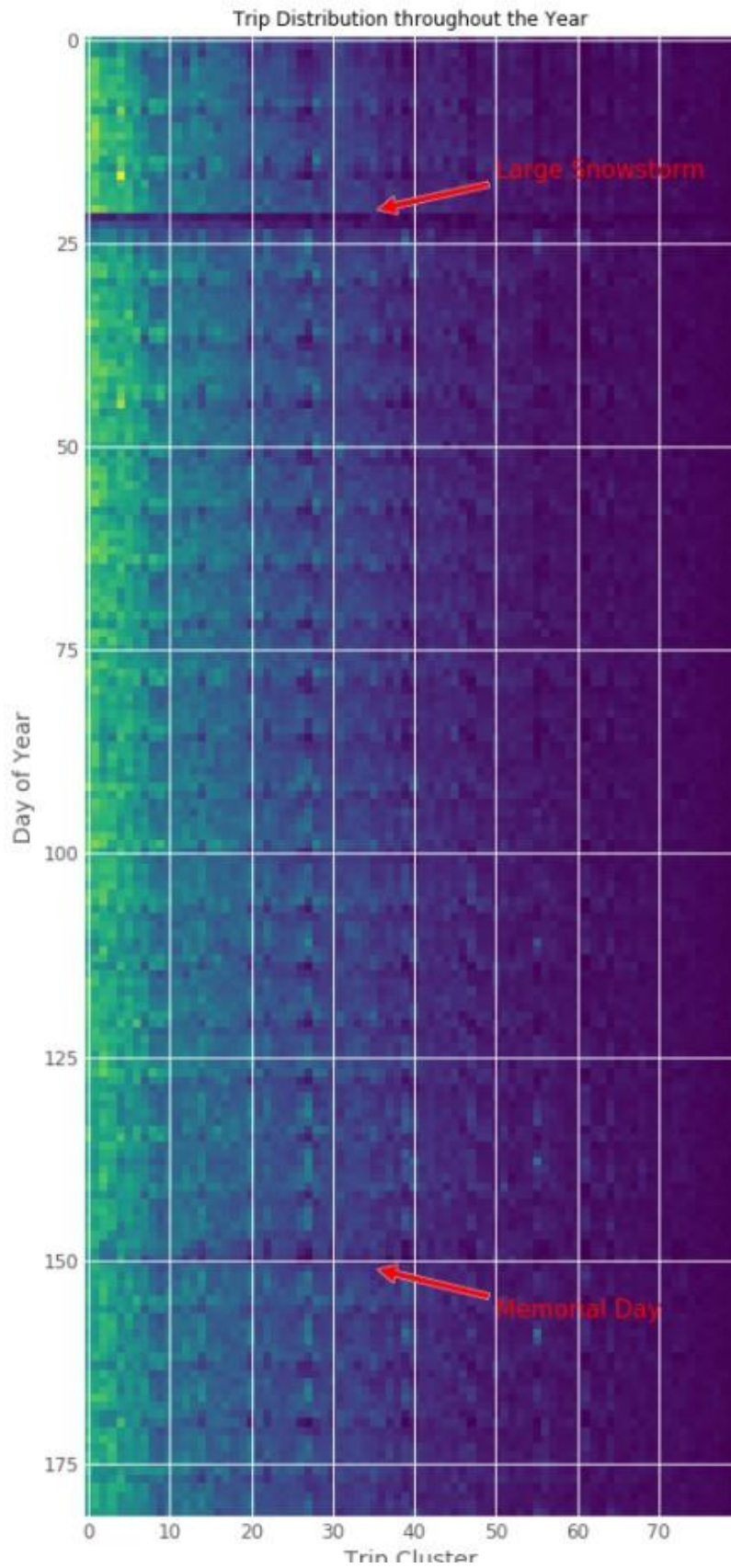
```

```
for i in topNegativeTripClusterInds:
    plt.arrow(srcImCoords[1][i],srcImCoords[0][i],
              dstImCoords[1][i]-srcImCoords[1][i], dstImCoords[0][i]-srcImCoords[0][i],
              edgecolor='b', facecolor='b',
              width=2.8,alpha=0.9,head_width=10.0,head_length=10.0,length_includes_head
              =True)
plt.title('PC3 major Trip deviations')
```

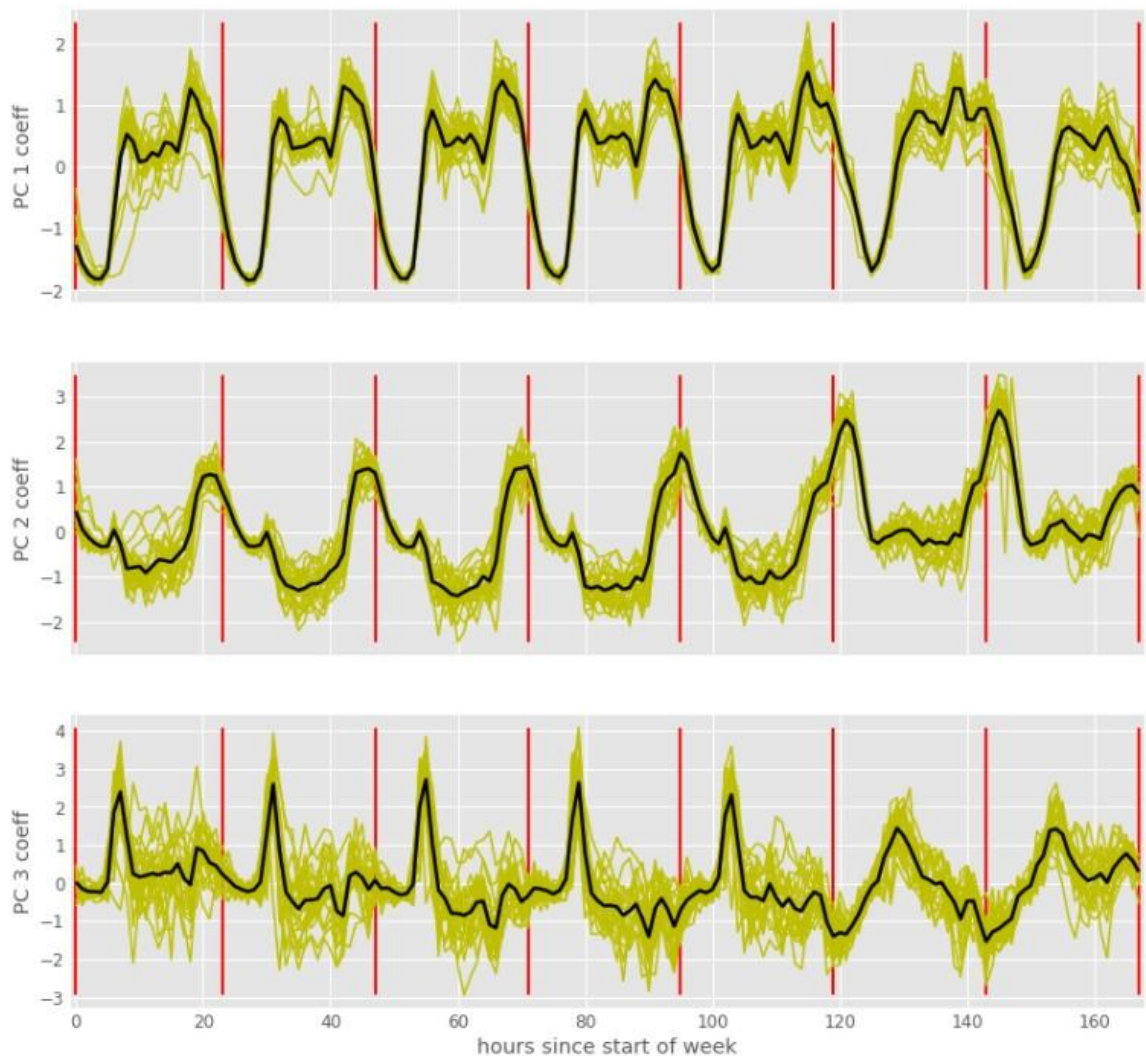
APPENDIX-II SAMPLE SHOTS



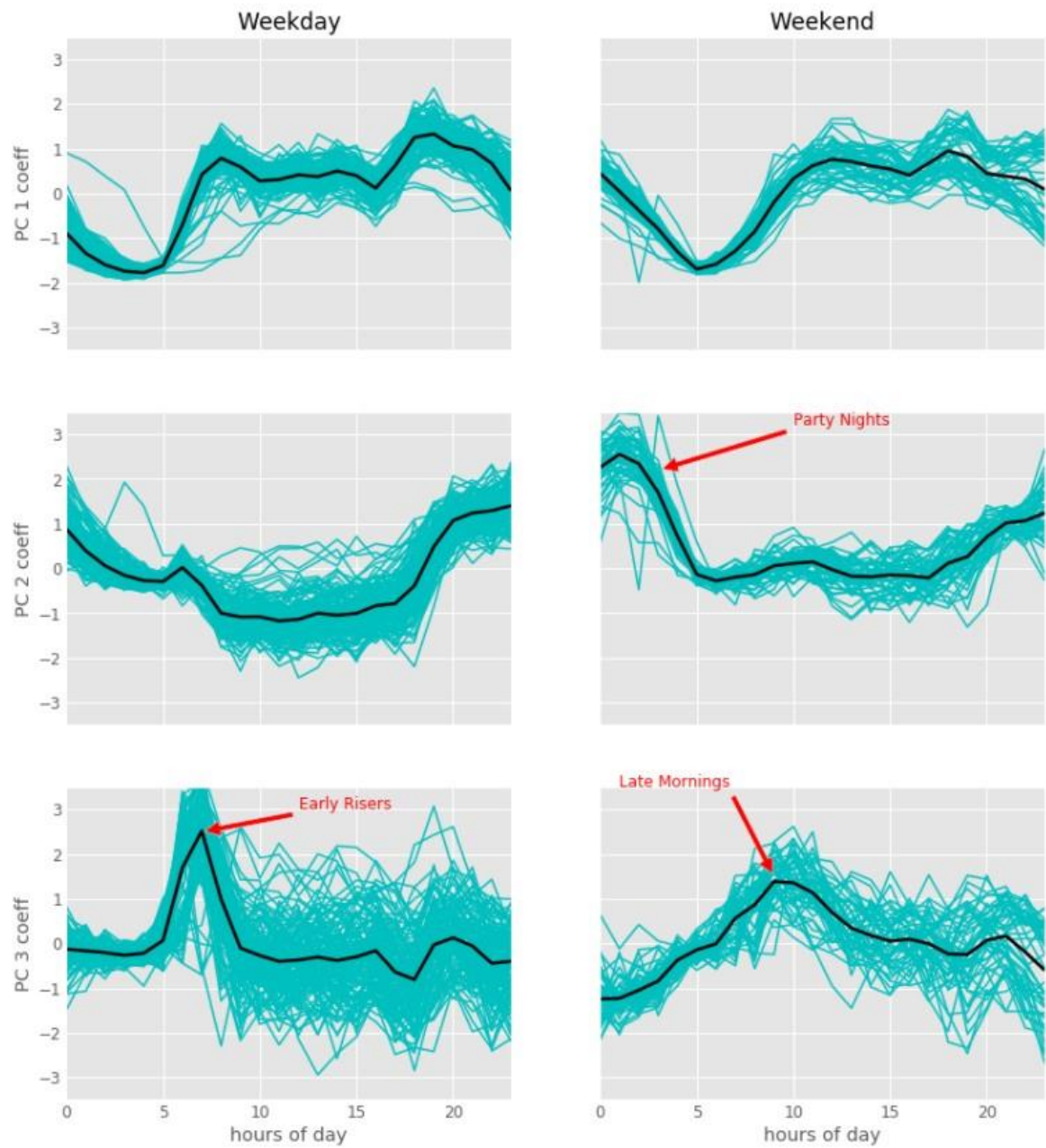




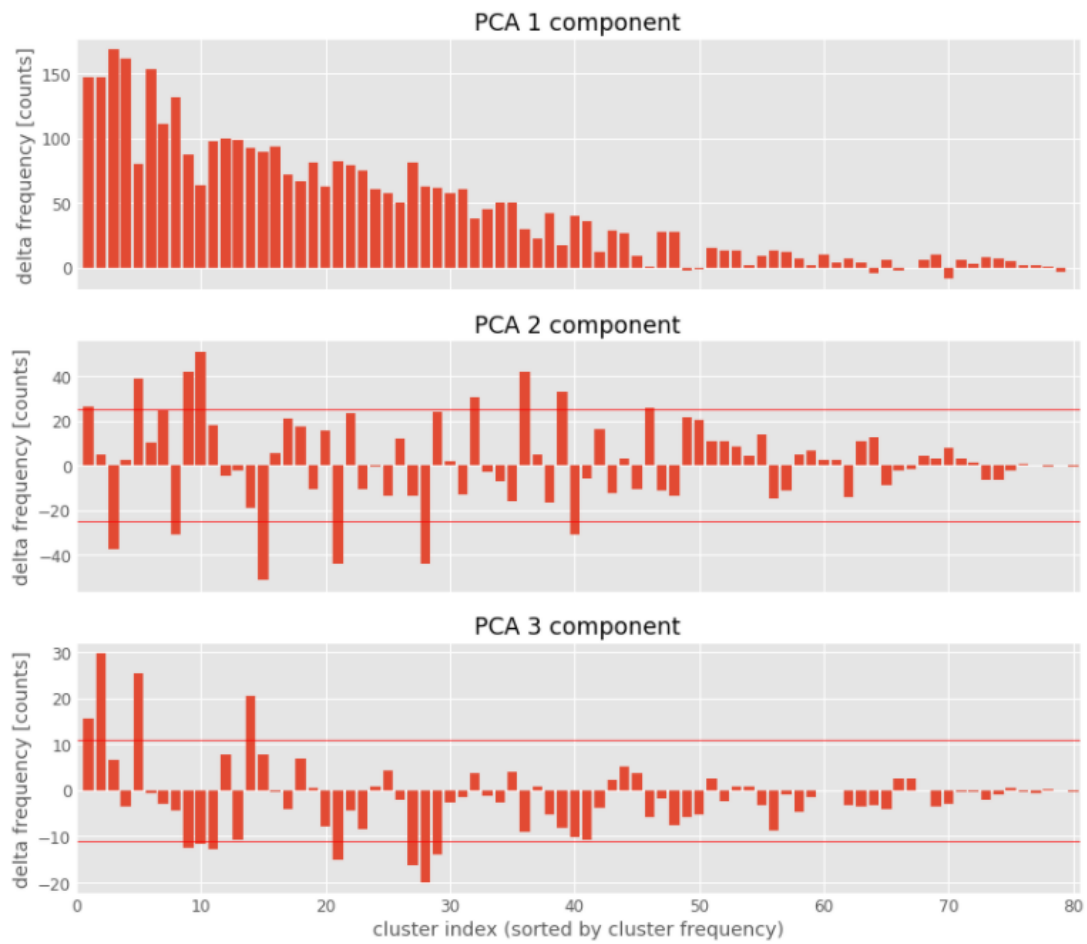
PCA coefficients during the Week

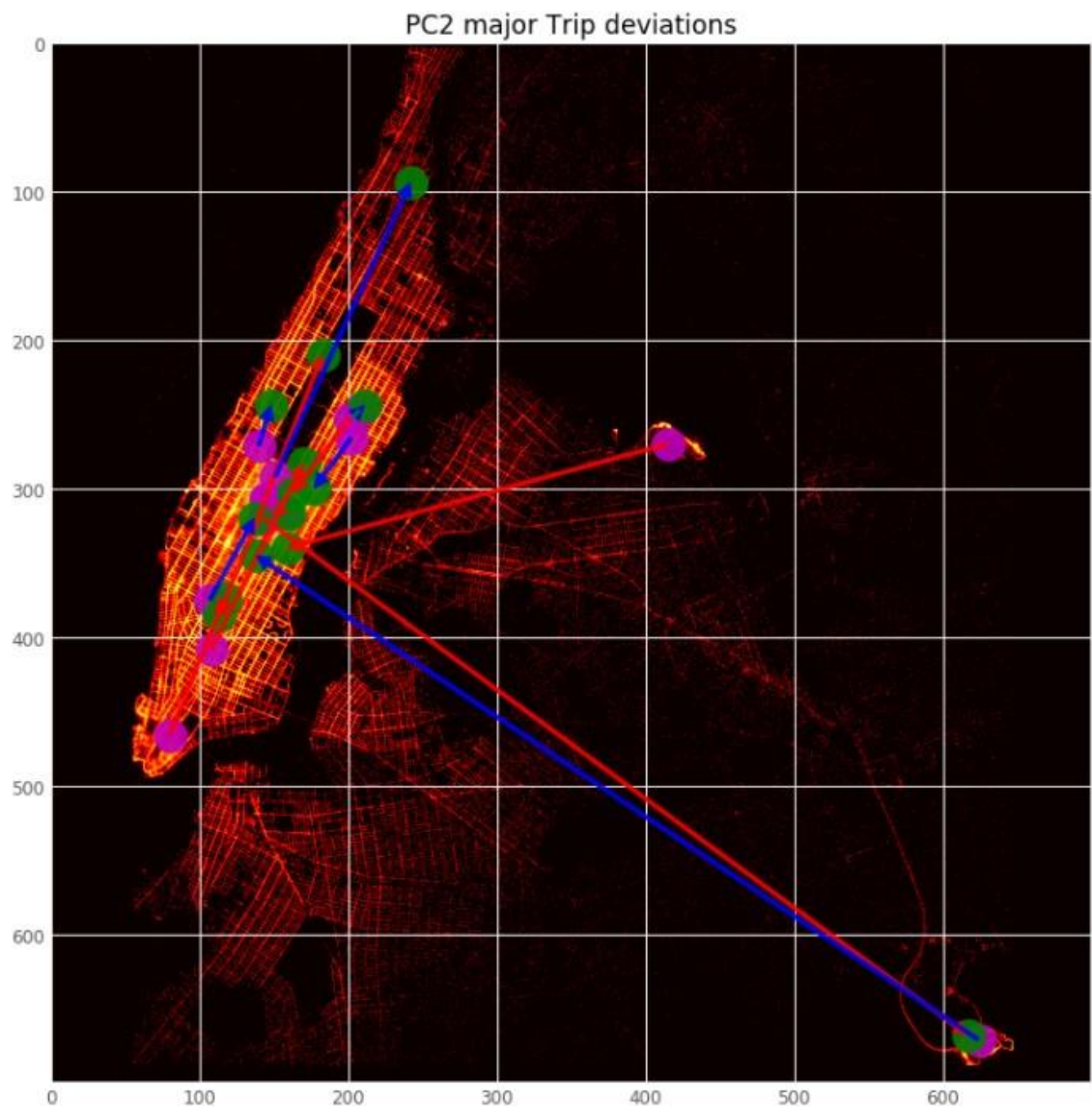


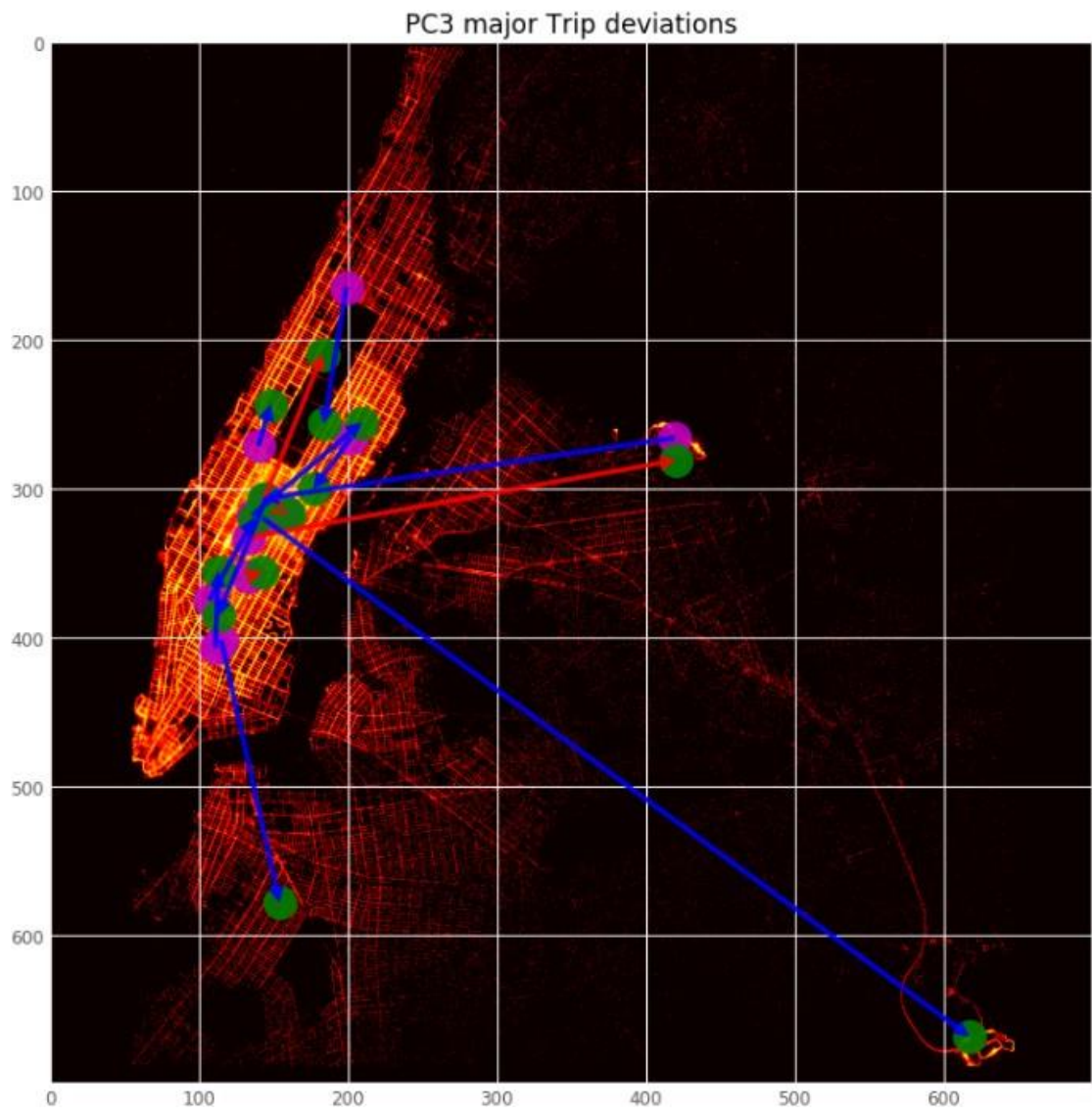
PCA coefficients for weekdays and weekends



Trip Distribution PCA Components







REFERENCE

- D. Zhang, L. Sun, B. Li, C. Chen, G. Pan, S. Li, and Z. Wu, "Understanding Taxi Service Strategies from Taxi GPS Traces," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, Feb. 2015, pp. 123–135, doi:10.1109/tits.2014.2328231.
- B. Cici, A. Markopoulou, E. Frias-Martinez, and N. Laoutaris, "Assessing the potential of ride-sharing using mobile and social data," in *Proc. ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 14)*, 2014, pp. 201–211, doi:10.1145/2632048.2632055.
- Y. Han, G. Wang, J. Yu, C. Liu, Z. Zhang, and M. Zhu, "A Service-Based Approach to Traffic Sensor Data Integration and Analysis to Support Community-Wide Green Commute in China," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, Sep. 2016, pp. 2648–2657, doi:10.1109/tits.2015.2498178
- P. Cheng, H. Xin, and L. Chen. Utility-aware ridesharing on road networks. In *ACM SIGMOD*, 2017.
- P. Goel, L. Kulik, and K. Ramamohanarao. Privacy-aware dynamic ride sharing. *ACM Transactions on Spatial Algorithms and Systems*, 2(1):1–41, 2016. ▶ W. He, K. Hwang, and D. Li. Intelligent carpool routing for urban ridesharing by mining GPS trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2286–2296, 2014.
- Y. Hou, W. Zhong, L. Su, K. Hulme, A. W. Sadek, and C. Qiao. TAsE: improving the efficiency of electric taxis with transfer-allowed rideshare. *IEEE Transactions on Vehicular Technology*, 65(12):9518–9528, 2016.
- Y. Huang, F. Bastani, R. Jin, and X. S. Wang. Large scale real-time ridesharing with service guarantee on road networks. *Proceedings of the VLDB Endowment*, 7(14):2017–2028, 2014.

K. Lin, R. Zhao, Z. Xu, and J. Zhou. Efficient large-scale fleet management via multi agent deep reinforcement learning. In ACM KDD, 2018.

Q. Lin, L. Dengt, J. Sun, and M. Chen. Optimal demand-aware ridesharing routing. In IEEE INFOCOM, 2018.

Z. Liu, Z. Li, M. Li, W. Xing, and D. Lu. Mining road network correlation for traffic estimation via compressive sensing. *IEEE Transactions on Intelligent Transportation Systems*, 17(7):1880–1893, 2016. ▶ Z. Liu, Z. Li, and K. Wu. UniTask: a unified task assignment design for mobile crowdsourcing-based urban sensing. *IEEE Internet of Things Journal*, 6(4):6629– 6641, 2019.